
# 中国研究生创新实践系列大赛

# “华为杯”第十六届中国研究生

# 数学建模竞赛

| | |
|---|---|
| 学　　校 | 国防科技大学 |
| 参赛队号 | F19910020011 |
| 队员姓名 | 1.王钊辉 |
| | 2.赵啸宇 |
| | 3.顾　伟 |

# 中国研究生创新实践系列大赛

# “华为杯”第十六届中国研究生

# 数学建模竞赛

题 目 多约束条件下智能飞行器航迹快速规划问题


## 摘 要：

复杂环境下智能飞行器航迹快速规划，可以确保飞行器合理运行的安全、稳定与高效。针对智能飞行器飞行过程中的误差校正问题，本文建立了智能飞行器的航迹规划模型。分别考虑了设有随机的校正点的快速规划的问题、有曲率约束的航迹规划问题、带有校正概率的航迹规划问题，并分别建立模型进行求解。

针对问题一，考虑最短航迹距离的基础上，同时考虑最小化经过的校正点数目。通过建立混合整数规划模型，通过设定目标的优先级，使之成为单目标优化问题，调用 Cplex 可在秒级时间内求得模型的最优解，航迹距离分别为 103516.89m 和 109342.28m。

针对问题二，在问题一的航迹规划基础上，考虑飞行器机动性能，加入因方向变换而产生的弧段航迹。在模型计算前，引入三维 Dubins 曲线，计算距离矩阵，可精确解得因转弯造成的航迹变化。通过 Cplex 求解，可以在秒级时间内求得问题的最优解，航迹距离分别为 103847.41m 与 109646.07m。为进一步提升求解时间，本文提出了优选校正点概念，在不影响解的优越性的情况下极大缩小 Dubins 矩阵规模，提升求解效率。

针对问题三，该问题加入了校正点的概率属性，首先通过收紧概率校正点的校正概率约束，求得题目数据在最差校正情况下的可行航迹，其成功到达终点的概率为 100%，航迹距离分别为 104827.37m 与 161650.41m。为提高模型的普适性，增强应对的其他情况的求解能力，本文提出了带校正概率的航迹规划概率计算方法，并结合蚁群算法，利用其全局搜索、正反馈的特点，提升解的质量。

针对复杂环境下智能飞行器航迹快速规划的问题，本文提出了航迹规划的混合整数规划模型，并根据问题二与问题三的特性，在模型中加入特定约束条件。三个问题均在较短时间内求得了最优解，符合智能飞行器航迹规划的稳定性与高效性的特点。模型实际应用价值高，针对大规模和复杂度比较高的问题，模型求解还需要进一步研究。



**关键词：**航迹规划 混合整数规划 三维 Dubins 曲线 蚁群算法

# 目录

# 一、问题重述

随着智能飞行器的迅速发展，亟需深入研究以保证智能飞行器圆满完成任务。由于系统结构限制，飞行器的定位系统无法对自身进行精准定位，一旦定位误差积累到一定程度可能导致任务失败。因此在飞行过程中对定位误差进行校正是智能飞行器航迹规划中一项重要任务。智能飞行器航迹规划是保证其圆满完成任务的重要技术支撑，也是智能飞行器任务规划系统的关键组成部分。本题目研究智能飞行器在系统定位精度限制下的三维航迹快速规划问题[1]。

航迹规划问题 RPP（Route Planning Problem）是指在综合考虑飞行器的机动性能、到达时间、飞行区域、以及外部环境等因素的前提下，在发射点和目标点之间为飞行器规划出一条最优的飞行航迹。航迹规划是智能飞行器圆满完成任务的重要保证，也是任务规划系统关键技术之一。随着现代信息技术的高速发展，航迹规划技术已经被广泛应用于各种军民无人飞行器任务规划系统中[2]。

考虑到智能飞行器受系统结构和机动性能影响，航路中可能存在不可飞航迹或尖角，造成智能飞行器不能按照预定的航路飞行，可能威胁到智能飞行器自身飞行安全，迫切需要寻找一种航迹的优化设计方法，使航路满足飞行器机动稳定性要求。本文结合智能飞行器航迹规划中其他的约束条件，建立了智能飞行器机动性能航迹校正模型和考虑校正概率的航迹规划模型，机动性能约束模型主要包括飞行垂直误差、飞行水平误差约束、转弯半径约束；考虑校正概率的航迹规划模型主要考虑天气、雷达等外部因素对校准概率的影响。

## 二、问题分析

根据主要任务可见，本题是按照分步设计的思路。在考虑规划智能飞行器航迹时，首先完成航迹校准任务以满足航迹安全稳定基本需求，在此基础上最小化航迹整体距离与校准点的使用数目，以提升智能飞行器的运行效率及经济效益。第二步从智能飞行器的系统性能及机动特性出发，尽可能的减小飞行器转弯对航迹带来的影响。最后将飞行过程中的外部环境纳入考虑，针对天气等不可控因素导致的在校正点无法完全校正的情况，对飞行器成功校准并抵达终点的航迹规划进行综合考量。

问题一是考虑航迹校准点的智能飞行器航迹规划。在设定起点与终点的情况下，确保飞行器抵达终点时的航迹误差处在规定范围内，优化目标为航迹长度尽量缩短和经过尽量少的航迹校准点。问题一中变量规模较小，考虑的约束复杂性较低，可以采用整数规划建模，并进行精确求解。

问题二是在问题一的基础上，考虑了飞行器的系统特性带来的转弯性能约束。无法瞬间改变速度方向，故无法按照问题一设定的航迹进行折线飞行。问题二在设计的航路点需要考虑三维空间内的弧线航迹及到达各校正点的速度方向。

问题三是在问题一的基础上，考虑外部飞行环境的动态变化，加入校正点的校正概率因素。校正点的校正概率只影响校正后的剩余误差，可先将剩余误差按照最差情况处理，寻找是否有可行解，可行解即为成功率 100%的解，即问题三最优解。若无可行解，可设计蚁群算法等智能优化算法，对校正点空间进行全局搜索，加入适当的规则使其更快收敛。

## 三、问题假设及符号说明

### 3.1 问题假设及说明

1、 航迹：出发点 A，各校正点及目的地 B 按规划到达顺序连接形成的折线或曲线
2、 未到达终点：到达终点 B 时，垂直误差或水平误差大于 $\theta$
3、 优选校准点：根据校准点与航迹 AB 的距离
4、 航迹校准：飞行器到达校正点即能够根据该位置的误差校正类型进行误差校正。校正垂直和水平误差的位置可根据地形在航迹规划前确定
5、 最小转弯半径：飞行器沿最大转弯方向转弯所构成航迹的圆的半径。
6、 速度方向：问题二中，每个点的速度方向，默认都指向终点 B。
7、 优化目标优先等级：
问题一中各目标的优先级为：最小化航迹长度>最小化经过航迹校正点数目。
问题二中各目标的优先级为：最小化航迹长度>最小化经过航迹校正点数目。
问题三中各目标的优先级为：最大化成功到达终点概率>最小化航迹长度>最小化经过航迹校正点数目。
8、 概率校正点：数据集里，问题三标记为 1 的点，即成功校正概率为 80%的校正点。

## 3.2 符号说明

表 1 本文所用符号说明

| 序号 | 标志 | 含义 |
|---|---|---|
| 1 | $x_{ij}$ | 飞行器是否经由 $i$ 点飞往 $j$ 点。 |
| 2 | $\Delta x_i$ | 飞行器到达 $i$ 点时水平误差（单位） |
| 3 | $\Delta y_i$ | 飞行器到达 $i$ 点时垂直误差（单位） |
| 4 | $\delta$ | 飞行器误差增量（单位/米） |
| 5 | $d_{ij}$ | 飞行器从 i 点到 j 点的距离 |
| 6 | C | 校正点集合 |
| 7 | V | 垂直校正点集合 |
| 8 | L | 水平校正点集合 |
| 9 | $\alpha_1$ | 垂直误差限制（垂直校正点，单位） |
| 10 | $\alpha_2$ | 水平误差限制（垂直校正点，单位） |
| 11 | $\beta_1$ | 垂直误差限制（水平校正点，单位） |
| 12 | $\beta_2$ | 水平误差限制（水平校正点，单位） |
| 13 | $\theta$ | 终点误差限制（垂直/水平，单位） |
| 14 | $l_i$ | $i$ 点的校正类型（0 水平校正，1 垂直校正） |
| 15 | $r$ | 最小转弯半径 |
| 16 | $X_1(X_{1x},X_{1y},X_{1z})$ | Dubins 曲线起点 |
| 17 | $X_2\ (X_{2x},X_{2y},X_{2z})$ | Dubins 曲线终点 |
| 18 | $V_1\ (V_{1x},V_{1y},V_{1z})$ | 起点速度向量 |
| 19 | $V_2\ (V_{2x},V_{2y},V_{2z})$ | 终点速度向量 |
| 20 | $w_i$ | 校正属性（0 普通校正点，1 概率校正点） |
| 21 | $p_i$ | i 点的校正概率 |
| 22 | $P_R$ | 路径 R 到达终点的概率 |
| 23 | $S$ | 校正结果能否到达终点 |

## 3.3 数据说明

本题共有两个数据文件，“附件 1：数据集 1-终稿”（以下简称数据集 1）与“附件 2：数据集 2-终稿” （以下简称数据集 2）。

数据集 1 中共有 611 个校正点，其中水平校正点 306 个、垂直校正点各 305 个。表 2 为数据集 1 的原始数据示例。

表 2 数据集 1 原始数据实例

| 编号 | X 坐标（m） | Y 坐标（m） | Z 坐标（m） | 校正点类型 | 第三问点标记 |
|---|---|---|---|---|---|
| 0 | 0.00 | 50000.00 | 5000.00 | A 点 | 0 |
| 1 | 33070.83 | 2789.48 | 5163.52 | 0 | 0 |
| 2 | 54832.89 | 49179.22 | 1448.30 | 1 | 1 |
| 3 | 77991.55 | 63982.18 | 5945.82 | 0 | 0 |
| … | … | … | … | … | … |
| 612 | 100000.00 | 59652.34 | 5022.00 | B 点 | 0 |

数据集 2 中共有 325 个校正点，其中水平校正点 167 个、垂直校正点各 158 个。表 3 为数据集 1 的原始数据示例。

表 3 数据集 2 原始数据实例

| 编号 | X 坐标（m） | Y 坐标（m） | Z 坐标（m） | 校正点类型 | 第三问点标记 |
|---|---|---|---|---|---|
| 0 | 0 | 50000 | 5000 | A 点 | 0 |
| 1 | 76009.85 | 9788.11 | 9121.90 | 1 | 0 |
| 2 | 2448.20 | 71599.88 | 1877.13 | 0 | 0 |
| … | … | … | … | … | … |
| 326 | 100000.00 | 74860.55 | 5499.61 | B 点 | 0 |

# 四、问题建模与求解

## 4.1 问题一建模与求解

### 4.1.1 航迹规划的混合整数规划模型

根据题目中的优化目标和约束要求，建立如下航迹规划的混合混合整数规划模型：

$$\text{Min} \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}d_{ij} \tag{1}$$

$$\text{Min} \sum_{i\in C}\sum_{j=1}^{n} x_{ij} \tag{2}$$

$$\sum_j x_{ij} \le 1, \forall i \tag{3}$$

$$\sum_i x_{ii} = 0, \forall i \tag{4}$$

$$\sum_{(i,j)} x_{ij} = \sum_{(j,i)} x_{ji}, i \ne 1, n \tag{5}$$

$$\sum_j x_{1j} = 1 \tag{6}$$

$$\sum_i x_{in} = 1 \tag{7}$$

$$\sum_j x_{nj} = 0 \tag{8}$$

$$\mathrm{M}\left(1 - x_{ij}\right) + \Delta x_j \ge \delta d_{ij} + \Delta x_i \cdot l_i, \forall i, j \tag{9}$$

$$\mathrm{M}\left(1 - x_{ij}\right) + \Delta y_j \ge \delta d_{ij} + \Delta y_i \cdot (1 - l_i), \forall i, j \tag{10}$$

$$\Delta x_i \le \alpha_2, i \in V \tag{11}$$

$$\Delta y_i \le \alpha_1, i \in V \tag{12}$$

$$\Delta x_i \le \beta_2, i \in L \tag{13}$$

$$\Delta y_i \le \beta_1, i \in L \tag{14}$$

$$\Delta x_n \le \theta, i \in L \tag{15}$$

$$\Delta y_n \le \theta, i \in L \tag{16}$$

其中，（1）表示最小化飞行器轨迹长度，（2）表示最小化经过校正区域进行校正的次数。（3）和（4）表示每个点最多只能经过一次，点本身不存在边，即飞行器不能在一个点上绕圈。约束（5）要求除出发点和终点外，航迹要实现节点的流平衡，即进入该节点的次数等于出于该节点的次数。约束（6）、（7）和（8）要求飞行器必须从出发点出发，最终在终点停止，并且不能从终点飞往其他节点。约束（9）和（10）为飞行器误差传递约束，飞行器在一个节点的水平和垂直误差等于前一个节点校正后误差加上从

前一节点到本节点的飞行距离造成的误差之和，当上一个节点为水平误差校正点时，校正后水平误差变为 0，垂直误差不变。当上一个节点为垂直误差校正点时，校正后垂直误差变为 0，水平误差不变。约束（11）至（16）要求飞行器在垂直误差校正点、水平误差校正点和终点要小于允许的垂直误差和水平误差最大值。

根据假设的目标优先级顺序，问题一的优化目标为在航迹长度尽可能小的情况下，经过校正区域次数尽可能少。通过加权求和可以将该多目标优化问题转化为单目标优化问题，优化目标为式（17），其中$k_1$远远大于$k_2(k_1>>k_2)$。

$$\text{Min } k_1 \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} d_{ij} + k_2 \sum_{i \in C} \sum_{j=1}^{n} x_{ij} \tag{17}$$

### 4.1.2 航迹规划的混合整数规划模型求解结果

在本问题的解决中，我们采用 JAVA 语言调用 CPLEX 的 API 进行航迹规划的混合整数规划模型求解。在配置为 CORE i7 7500U，主频 2.7GHZ，4GB 内存的电脑上，数据集 1 可以由 CPLEX 在 12.979s 内求得最优解。

经过的校正点顺序依次为 0，503，200，80，237，170，278，369，214，397，612，总距离为 103516.89m。数据集 1 航迹共经过 11 个点，除起点与终点外，经过 9 个航迹校正点，其中 4 个水平校正点。5 个垂直校正点。

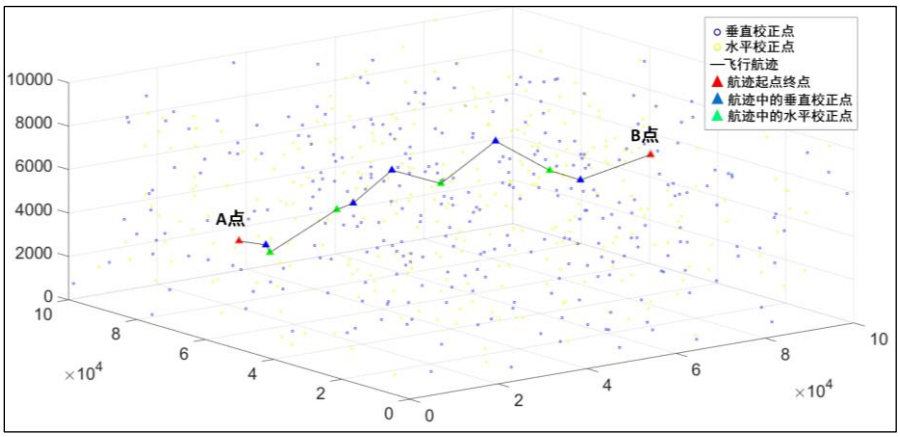



图 1 问题一数据集 1 求解结果三维航迹图

表 4 问题一数据集 1 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **503** | 13.39 | 13.39 | 11（垂直校正点） |
| **200** | 0.87 | 14.25 | 01（水平校正点） |
| **80** | 16.61 | 15.75 | 01（水平校正点） |
| **237** | 21.24 | 4.63 | 11（垂直校正点） |
| **170** | 7.69 | 12.32 | 11（垂直校正点） |
| **278** | 10.46 | 22.77 | 01（水平校正点） |
| **369** | 21.89 | 11.44 | 11（垂直校正点） |
| **214** | 13.31 | 24.75 | 01（水平校正点） |

| | | | |
|---|---|---|---|
| **397** | 22.33 | 9.02 | 11（垂直校正点） |
| **612** | 16.97 | 25.99 | 终点 B |

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正点允许的最大水平误差与垂直误差（见图 2、图 3），抵达终点后的垂直误差和水平误差均不超过 $\theta$（30 米），满足题目要求。

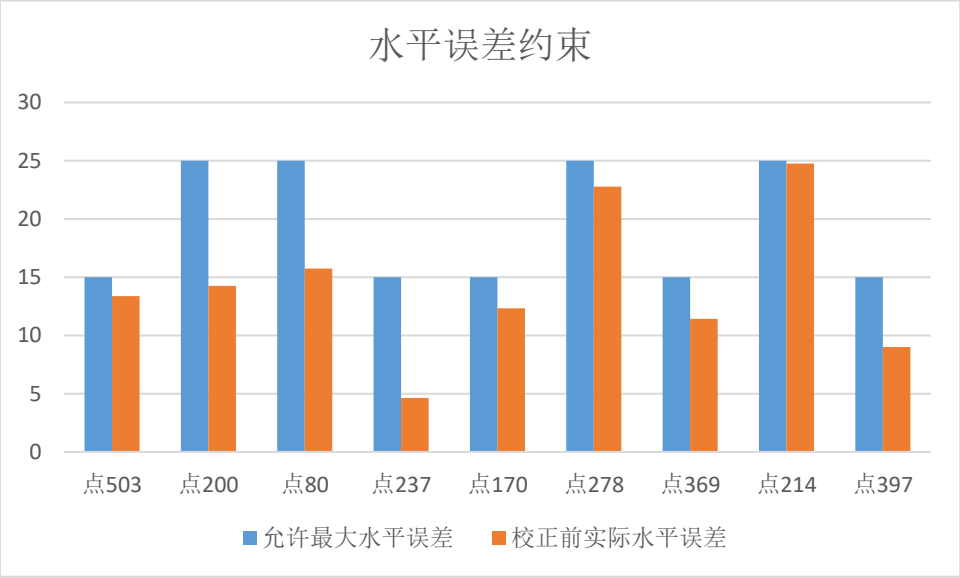



图 2 问题一数据集 1 各校正点前水平误差约束检查

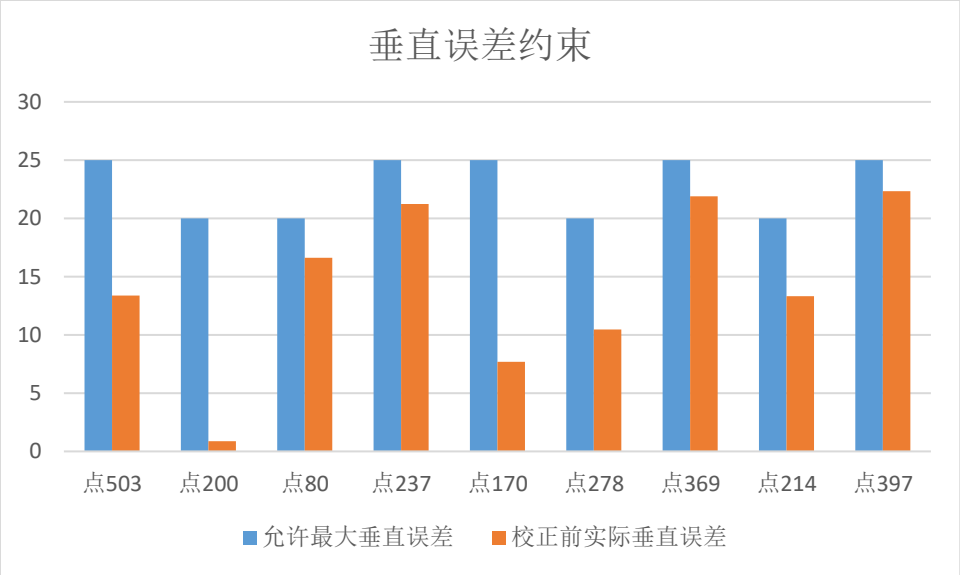



图 3 问题一数据集 1 各校正点前垂直误差约束检查

在数据集 2 的解决中，我们采用与数据集 1 相同配置，可以在 4.972s 内求得航迹规划的混合整数规划模型的最优解。数据集 2 航迹共经过 14 个点，经过的各点顺序依次为 0，163，114，8，309，305，123，45，160，92，93，61，292，326，总距离为

109342.28m。除起点与终点外，经过 12 经过个航迹校正点，其中 6 个水平校正点、6 个垂直校正点。

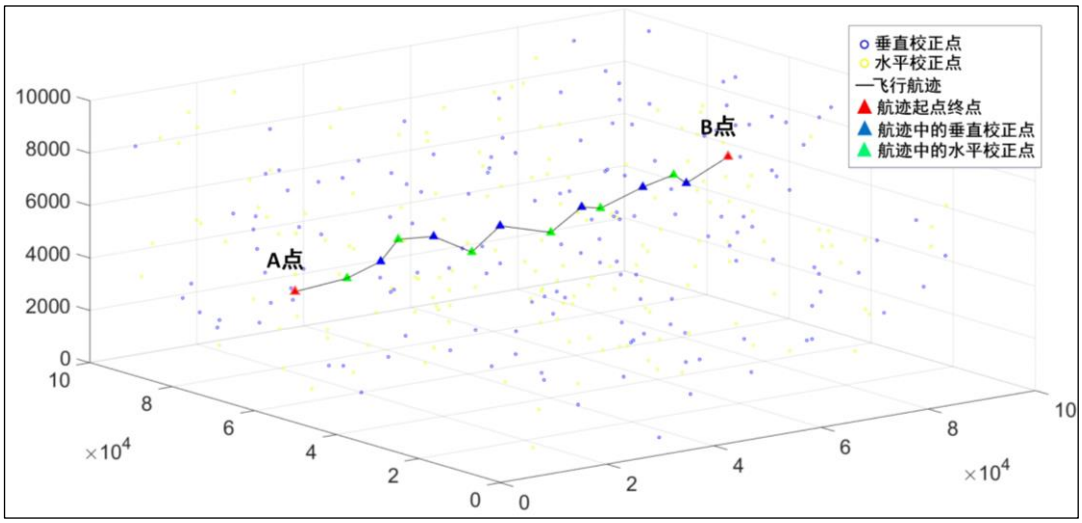



图 4 问题一数据集 2 求解结果三维航迹图

表 5 问题一数据集 2 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **163** | 13.29 | 13.29 | 01（水平校正点） |
| **114** | 18.62 | 5.33 | 11（垂直校正点） |
| **8** | 13.92 | 19.26 | 01（水平校正点） |
| **309** | 19.45 | 5.52 | 11（垂直校正点） |
| **305** | 5.97 | 11.49 | 01（水平校正点） |
| **123** | 15.17 | 9.20 | 11（垂直校正点） |
| **45** | 10.01 | 19.21 | 01（水平校正点） |
| **160** | 17.49 | 7.49 | 11（垂直校正点） |
| **92** | 5.78 | 13.26 | 01（水平校正点） |
| **93** | 15.26 | 9.48 | 11（垂直校正点） |
| **61** | 9.83 | 19.32 | 01（水平校正点） |
| **292** | 16.39 | 6.55 | 11（垂直校正点） |
| **326** | 6.96 | 13.51 | 终点 B |

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正点允许的最大垂直误差与水平误差（见图 5、图 6），抵达终点后的垂直误差和水平误差均不超过 $\theta$（20 米），满足题目要求。

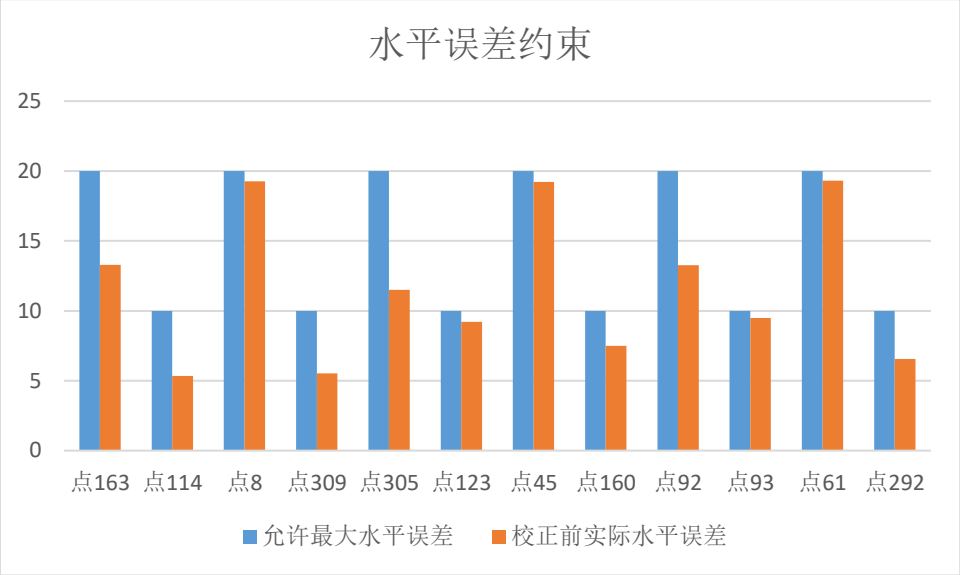



图 5 问题一数据集 2 各校正点前水平误差约束检查

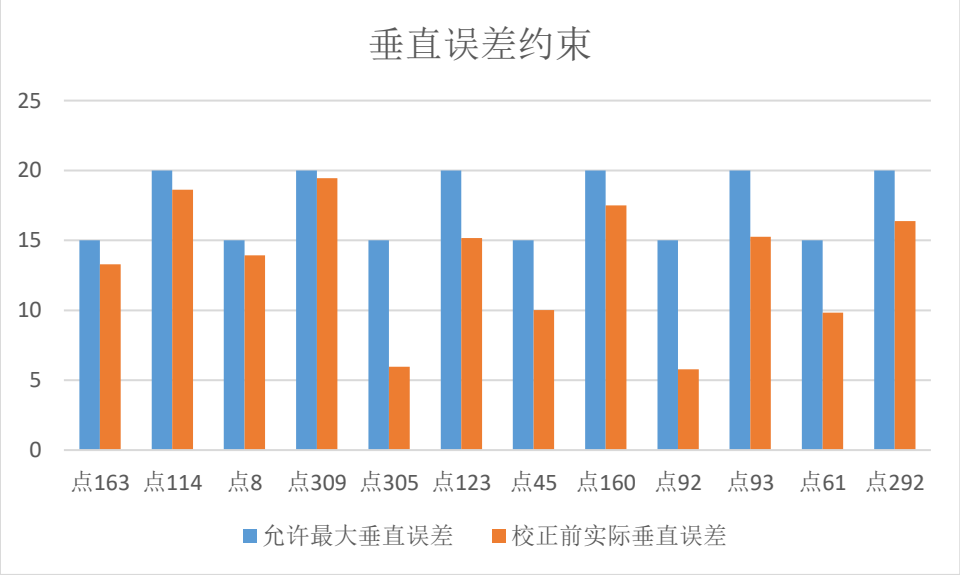



图 6 问题一数据集 2 各校正点前垂直误差约束检查

## 4. 2 问题二建模与求解

### 4.2.1 考虑最小转弯半径的航迹规划模型

问题二在问题一的基础上，加入了转弯动作的实现，具体表现在智能飞行器的运动过程中，无法实时改变速度方向，而需要进行转弯机动。点与点之间的距离$d_{ij}$计算时需要计算方向变换带来的额外的弧线部分长度。

问题二模型主体部分与问题一相同，可参考公式（1）-（17）。在 4.2.2 节中，我们将具体解释如何计算问题二中弧线部分的长度。

### 4.2.2 考虑最小转弯半径的三维 Dubins 最优路径

在二维平面上，Dubins[3]几何地给出了固定初始和最终位置及速度方向的光滑最短

路径。针对问题二中带曲率的三维最短路径，Sikha Hota[4]给出了在给定初始和最终构型参数的情况下，若两点间距足够远（距离大于四倍曲率半径），在给定曲率约束下，计算三维空间中的最优路径的方法。

在表示路径时，C 表示弧线，S 表示直线。对于这种情况，路径将是 CSC 类型，Sikha 使用与 2D 中 Dubins 曲线相同的原理来构造它，这种几何方法在短时间内生成最优路径。该方法简单，计算量小，可在转弯半径受限的飞行器上实现。

在二维平面上，已经证明固定初始位置和最终位置，并且固定两个方向的最短路径由 Dubins 集的三个连续路径段 D 组成，其中包括六类路径 D={LSL，RSR，RSL，LSR，RLR，LRL}（参见图 7）。在二维 Dubins 曲线中，按照最小曲率半径进行的向左/向右转弯分别以 L/R 标识，切线之间的直线用 S 标识。已经证明，在这类长路径问题中，CSC 的路径优于 CCC[3]。

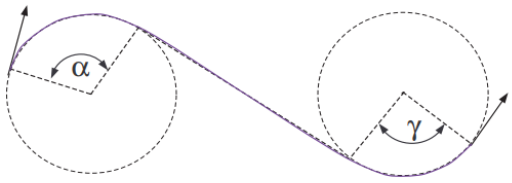



图 7 CSC 的 Dubins 路径

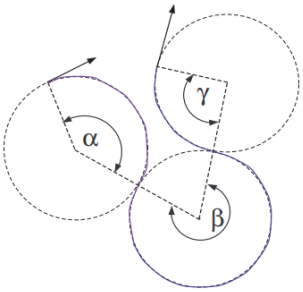



图 8 CCC 的 Dubins 路径

下图为 Sikha 提出的三维空间 Dubins 曲线计算示例图。其中输入参数为起点 $X_1$，终点 $X_2$，起点速度向量 $V_1$，终点速度向量 $V_2$ 及最小转弯半径 $r$。

表 6 Dubins 曲线计算输入

| 序号 | 标志 | 含义 |
|---|---|---|
| 1 | $r$ | 最小转弯半径 |
| 2 | $X_1(X_{1x},X_{1y},X_{1z})$ | Dubins 曲线起点 |
| 3 | $X_2\ (X_{2x},X_{2y},X_{2z})$ | Dubins 曲线终点 |
| 4 | $V_1\ (V_{1x},V_{1y},V_{1z})$ | 起点速度向量 |
| 5 | $V_2\ (V_{2x},V_{2y},V_{2z})$ | 终点速度向量 |

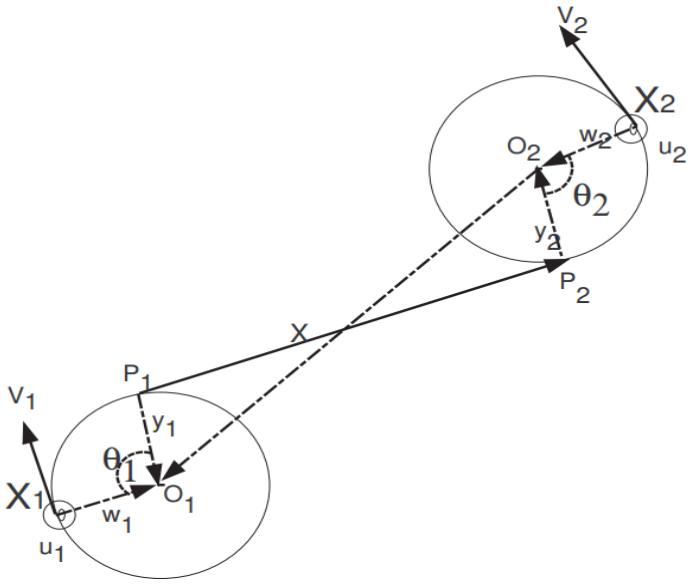



图 9 三维空间的 Dubins 曲线计算示例图

按照最小曲率半径转弯的初始曲线和最终曲线处于不同的平面，两端弧线之间 XX 的直线部分是两个平面的交线，让这两个平面的公共向量都是$X(X_x, X_y, X_z)$，则单位向量为

$$x = \frac{X}{||X||} \tag{18}$$

$X$ 可由以下公式解出

$$X = (X_2 + rw_2 - ry_2) - (X_1 + rw_1 - ry_1) \tag{19}$$

具体推导过程见附录，在此只列出求解过程中的方程组。

$$cos\theta_1 = v_1 \cdot x \tag{20}$$

$$cos\theta_2 = v_2 \cdot x \tag{21}$$

$$X_{2x} - X_{1x} = X_x + rx_x[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2x}\tan\frac{\theta_2}{2} + v_{1x}\tan\frac{\theta_1}{2}] \tag{22}$$

$$X_{2y} - X_{1y} = X_y + rx_y[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2y}\tan\frac{\theta_2}{2} + v_{1y}\tan\frac{\theta_1}{2}] \tag{23}$$

$$X_{2z} - X_{1z} = X_z + rx_z[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2z}\tan\frac{\theta_2}{2} + v_{1y}\tan\frac{\theta_1}{2}] \tag{24}$$

最终的 Dubins 曲线由三段组成，第一段弧长、直线长度、第二段弧长计算公式分别为(25)~(27)，各符号项具体含义参见附录。

$$rcos^{-1}(w_1 \cdot y_1) \tag{25}$$

$$(X_2 + rw_2 - ry_2) - (X_1 + rw_1 - ry_1) \tag{26}$$

$$rcos^{-1}(w_2 \cdot y_2) \tag{27}$$

经由上述方程，可以解得四种类型的 CSC 路径，下面给出了完整的方程组

$$X = (X_2 - X_1) \mp r(x+v_1)\frac{tan\theta_1}{2} \mp r(x+v_2)\frac{tan\theta_2}{2} \tag{28}$$

因四类 CSC 路径均为可行路径，因此在应用中，我们选择其中最短的路径。

### 4.2.3 考虑最小转弯半径的规划模型求解结果

Dubins 曲线的入射速度方向与出射速度方向均为决策变量，为简化计算量，提升求解效率，本文假设所有点的速度均朝向终点方向。

预先求得 Dubins 距离矩阵后，数据集 1 可以由 CPLEX 求得最优解。经过的校正点顺序依次为 0，503，200，80，237，170，278，369，214，397，612，总距离为 103847.41m。数据集 1 航迹共经过 11 个点，除起点与终点外，经过 9 个航迹校正点，其中 4 个水平校正点。5 个垂直校正点。

因问题二的转弯半径较小，经过与问题一的结果对比，问题二结果的距离增量不明显。采用了三维 Dubins 曲线后，弧线段长度相较原有直线部分并未增加很多。以图 10 的转弯平面示意图为例，假设航空器由点 $P$ 至点 $Q$，转角为 $\theta$，转弯半径为 $r$。则

$$PQ = 2r\,sin\frac{\theta}{2} \tag{29}$$

$$\widehat{PQ} = \theta r \tag{30}$$

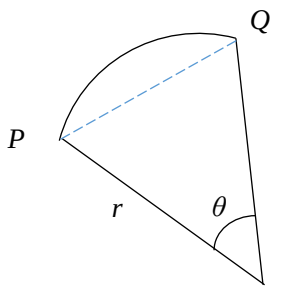



图 10 转弯平面的弧长计算示意图

因采用 Dubins 曲线，故转弯半径为 200 米。若转弯角度为 60 度，则 PQ 为 200 米，$\widehat{PQ}$约为 209 米，且由于弧线段长度在整个航迹长度中长度中占比远小于直线线段长度，故问题二的航迹距离总长度与问题一相差不大。

图 11 为问题二数据集 1 航迹及校正点图，在航迹中途经过的校正点上，各有与该点切入速度与切出速度的方向相切，由最小转弯半径构成的两个圆。在 A、B 分别有切出速度圆与切入速度所在方向相切，由最小转弯半径组成的一个圆。

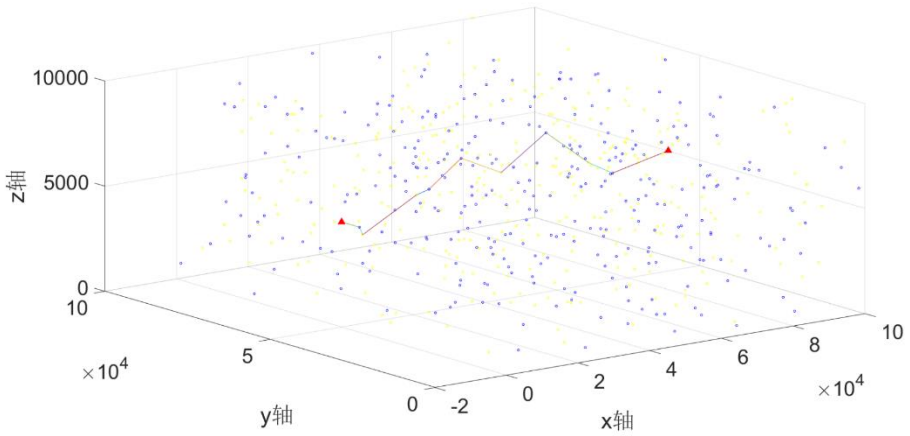



图 11 问题二数据集 1 航迹及校正点图（航迹间衔接为两个半径为 200 的圆弧）

表 7 问题二数据集 1 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **503** | 13.41 | 13.41 | 11（垂直校正点） |
| **200** | 1.11 | 14.51 | 01（水平校正点） |
| **80** | 16.90 | 15.79 | 01（水平校正点） |
| **237** | 21.53 | 4.63 | 11（垂直校正点） |
| **170** | 7.69 | 12.33 | 11（垂直校正点） |
| **278** | 10.46 | 22.78 | 01（水平校正点） |
| **369** | 21.90 | 11.44 | 11（垂直校正点） |
| **214** | 13.31 | 24.76 | 01（水平校正点） |
| **397** | 22.34 | 9.03 | 11（垂直校正点） |
| **612** | 16.97 | 26.00 | 终点 B |

表 7 为数据集 1 的结果中，各段弧线航迹对应的切点坐标，切点序号即对应航迹弧段序号，出切点坐标为航迹起点部分弧段，入切点坐标对应航迹终点部分弧段。

表 8 问题二数据集 1 切点坐标表

| 切点序号 | 出切点坐标 | | | 入切点坐标 | | |
|---|---|---|---|---|---|---|
| **1** | 31.65 | 50000.87 | 4998.75 | 11370.08 | 56973.14 | 4098.65 |
| **2** | 11497.20 | 56969.44 | 4070.69 | 12037.02 | 56743.87 | 3760.57 |
| **3** | 12177.01 | 56743.26 | 3736.47 | 27772.12 | 57540.78 | 5120.79 |
| **4** | 27819.12 | 57543.90 | 5123.95 | 32303.01 | 58618.74 | 5213.88 |
| **5** | 32356.16 | 58617.06 | 5218.57 | 39631.37 | 56895.57 | 6545.77 |
| **6** | 39723.25 | 56893.57 | 6546.64 | 50020.41 | 56062.83 | 5610.62 |
| **7** | 50115.13 | 56072.43 | 5604.63 | 61259.21 | 54644.97 | 7236.35 |
| **8** | 61310.27 | 54655.81 | 7233.27 | 74407.68 | 55885.70 | 5363.18 |
| **9** | 74515.36 | 55927.57 | 5353.17 | 82989.58 | 58143.24 | 4521.93 |
| **10** | 83113.16 | 58185.72 | 4514.26 | 99985.34 | 59651.57 | 5021.36 |

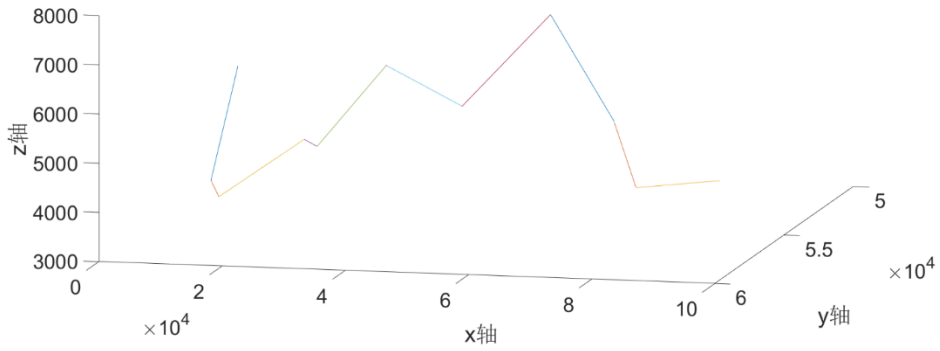



图 12 问题二数据集 1 航迹图（航迹间衔接为半径为 200 的圆弧）

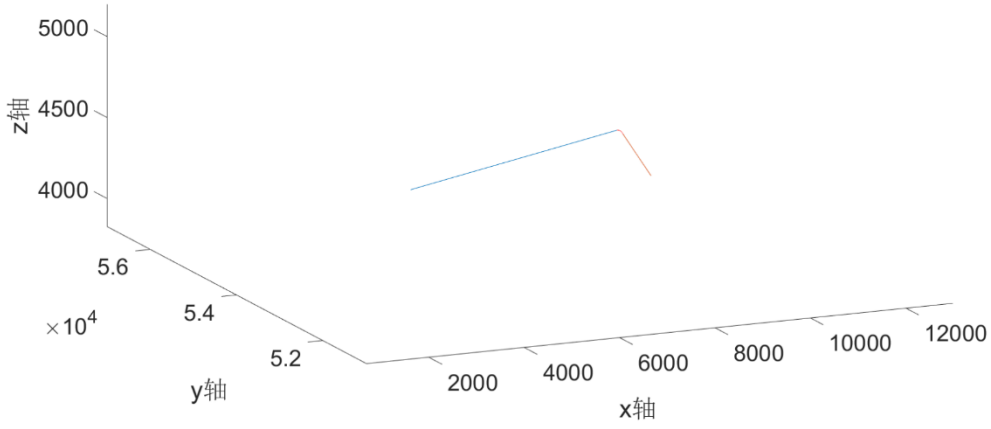



图 13 问题二数据集 1 前两段航迹图（点 A-点 503-点 200）

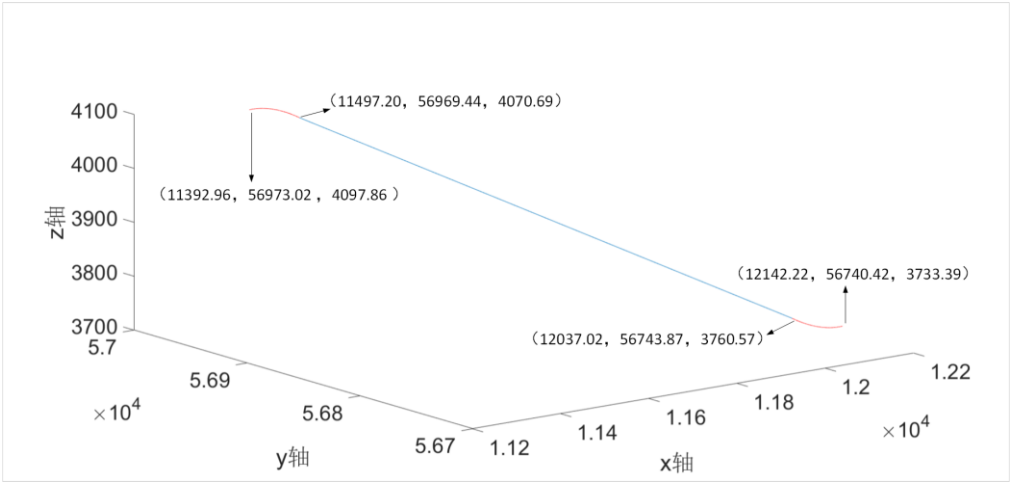



图 14 问题二数据集 1 单独显示第二段航迹（点 503-点 200）

图 15 为切入弧线与切出弧线示意图，在此我们选择点 170，图中两个红色圆形分别为点 237 出发切入 170 时曲率半径对应的圆与点 170 出发至 278 时曲率半径对应的圆。

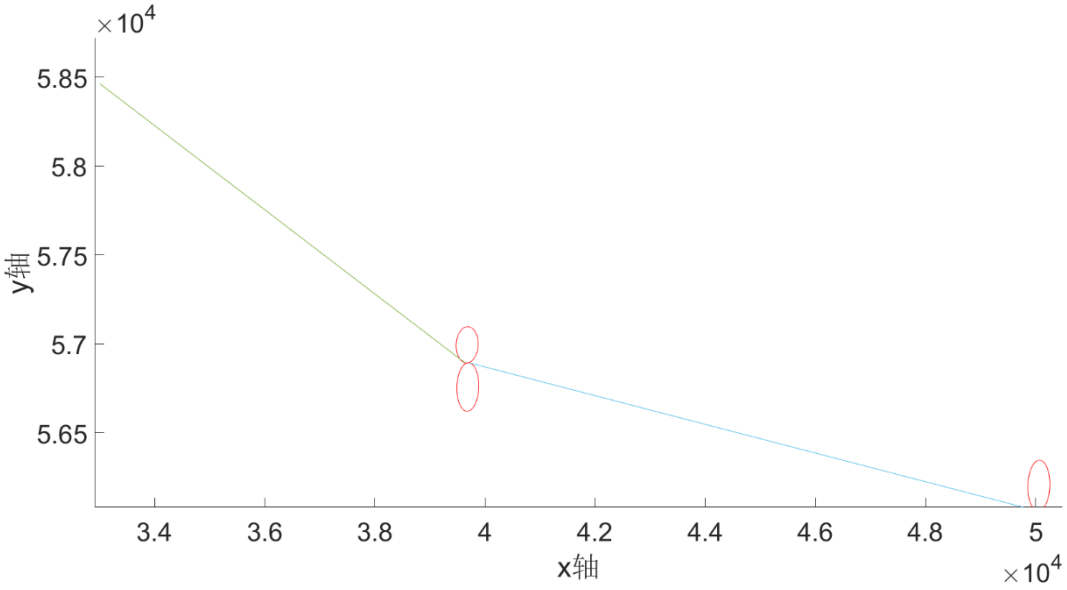



图 15 数据集 1 第六航迹点（点 170）的切入弧线与切出弧线示意图

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正点允许的最大垂直误差与水平误差，抵达终点后的垂直误差和水平误差均不超过 $\theta$（30 米），满足题目要求。

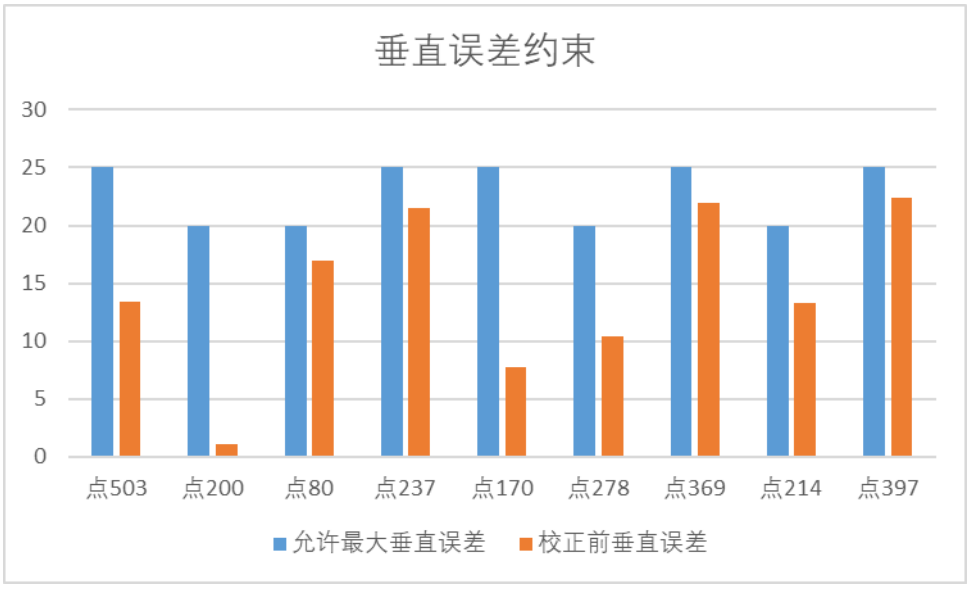



图 16 问题二数据集 1 各校正点前垂直误差约束检查

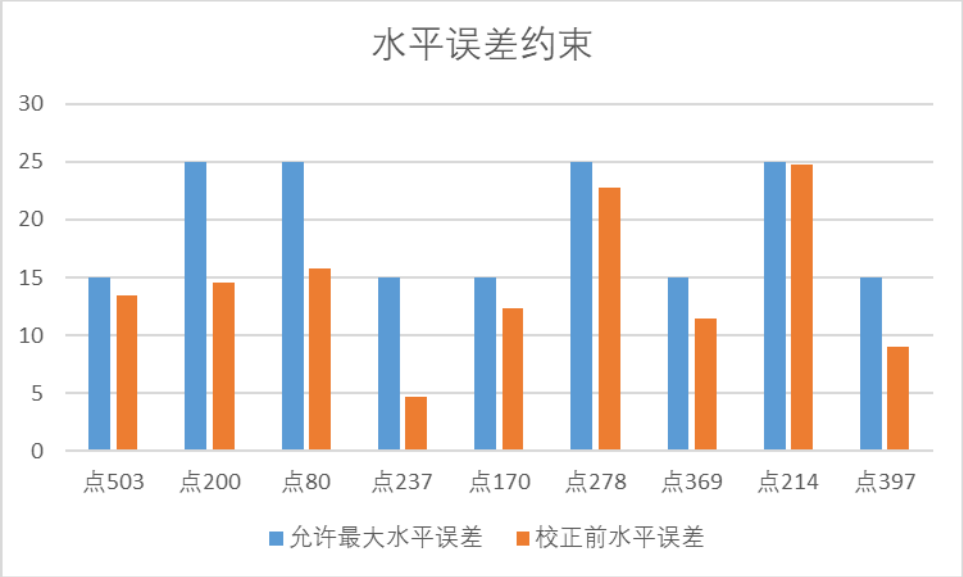



图 17 问题二数据集 1 各校正点前水平误差约束检查

预先求得 Dubins 距离矩阵后，数据集 2 可以由 CPLEX 求得最优解。经过的校正点顺序依次为 0，503，200，80，237，170，278，369，214，397，612，总距离为 109646.07m。数据集 2 航迹共经过 14 个点，经过的各点顺序依次为 0，163，114，8，309，305，123，45，160，92，93，61，292，326，总距离为 109342.28m。除起点与终点外，经过 12 经过个航迹校正点，其中 6 个水平校正点、6 个垂直校正点。

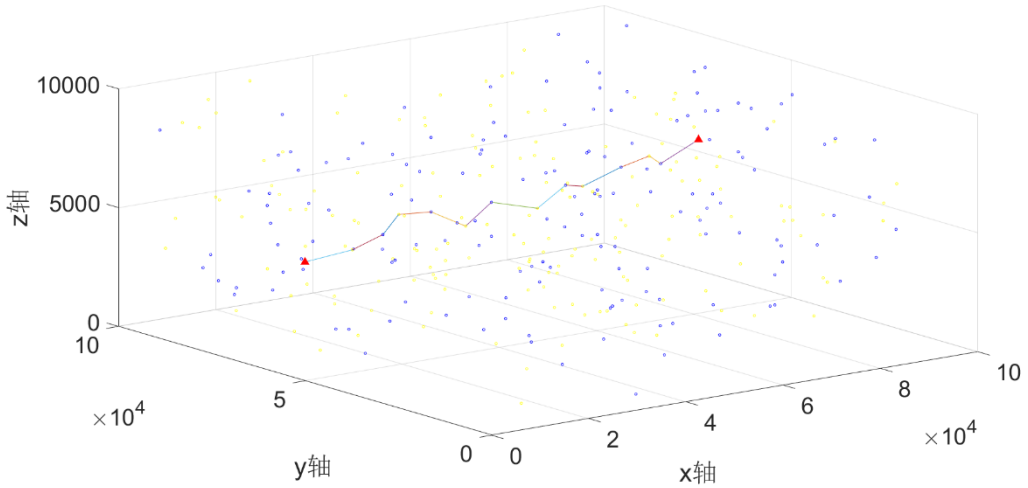



图 18 数据集 2 航迹及校正点图（航迹间衔接为两个半径为 200 的圆弧）

表 9 问题二数据集 2 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **163** | 13.29 | 13.29 | 01（水平校正点） |
| **114** | 18.65 | 5.36 | 11（垂直校正点） |

| | | | |
|---|---|---|---|
| **8** | 13.93 | 19.29 | 01（水平校正点） |
| **309** | 19.46 | 5.53 | 11（垂直校正点） |
| **305** | 6.01 | 11.55 | 01（水平校正点） |
| **123** | 15.22 | 9.21 | 11（垂直校正点） |
| **45** | 10.01 | 19.22 | 01（水平校正点） |
| **160** | 17.51 | 7.50 | 11（垂直校正点） |
| **92** | 5.78 | 13.28 | 01（水平校正点） |
| **93** | 15.27 | 9.49 | 11（垂直校正点） |
| **61** | 9.84 | 19.32 | 01（水平校正点） |
| **292** | 16.48 | 6.64 | 11（垂直校正点） |
| **326** | 7.06 | 13.70 | 终点 B |

表 10 为数据集 1 的结果中，各段弧线航迹对应的切点坐标，切点序号即对应航迹弧段序号，出切点坐标为航迹起点部分弧段，入切点坐标对应航迹终点部分弧段。

表 10 问题二数据集 2 切点坐标表

| 切点序号 | 出切点坐标 | | | 入切点坐标 | | |
|---|---|---|---|---|---|---|
| **1** | 10.92 | 50002.48 | 4999.84 | 12702.66 | 53862.88 | 4887.69 |
| **2** | 12761.89 | 53871.22 | 4890.44 | 17724.88 | 52280.16 | 5422.08 |
| **3** | 17822.11 | 52304.76 | 5425.42 | 28106.62 | 61538.99 | 5480.50 |
| **4** | 28188.69 | 61565.49 | 5480.85 | 33355.52 | 59706.78 | 5490.92 |
| **5** | 33401.18 | 59716.42 | 5489.03 | 38896.22 | 57656.83 | 4794.53 |
| **6** | 39002.43 | 57698.42 | 4796.90 | 47162.58 | 61568.60 | 5318.30 |
| **7** | 47280.90 | 61610.31 | 5320.82 | 57148.58 | 62242.00 | 4690.72 |
| **8** | 57272.14 | 62258.68 | 4692.51 | 64347.66 | 64117.50 | 5317.55 |
| **9** | 64485.03 | 64158.29 | 5316.82 | 69635.41 | 66399.23 | 4989.13 |
| **10** | 69762.86 | 66440.39 | 4986.80 | 78941.11 | 68239.58 | 5379.59 |
| **11** | 79071.33 | 68283.96 | 5380.77 | 87847.66 | 72333.47 | 5346.76 |
| **12** | 87996.60 | 72376.52 | 5346.70 | 93188.84 | 76193.28 | 4663.89 |
| **13** | 93243.92 | 76185.39 | 4668.69 | 99974.55 | 74863.83 | 5496.48 |

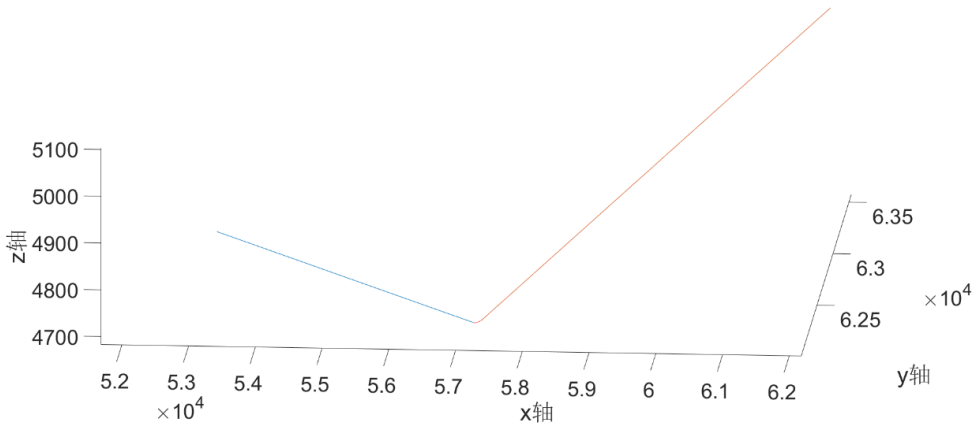



图 19 数据集 2 第八航迹点（点 45）的航迹示意图

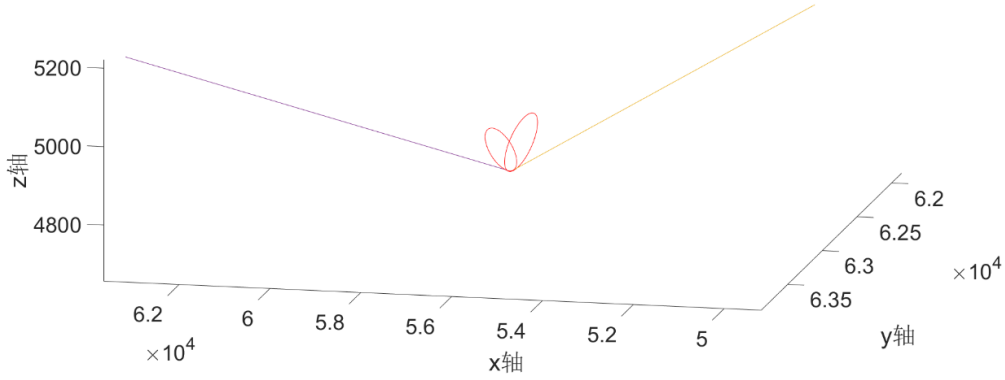



图 20 数据集 2 第八航迹点（点 45）的切入弧线与切出弧线所在圆示意图

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正点允许的最大垂直误差与水平误差（见下图），抵达终点的误差范围也不超过 $\theta$。

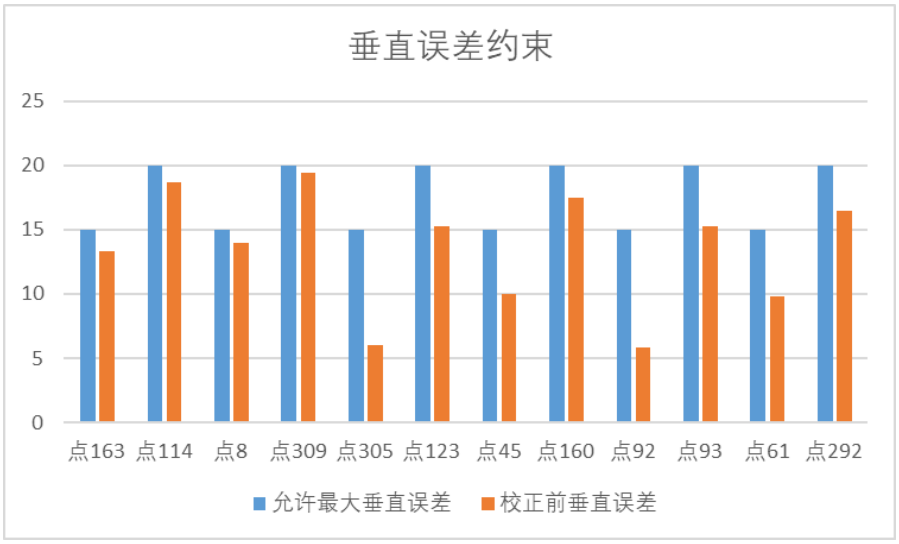



图 21 问题二数据集 2 各校正点前水平误差约束检查

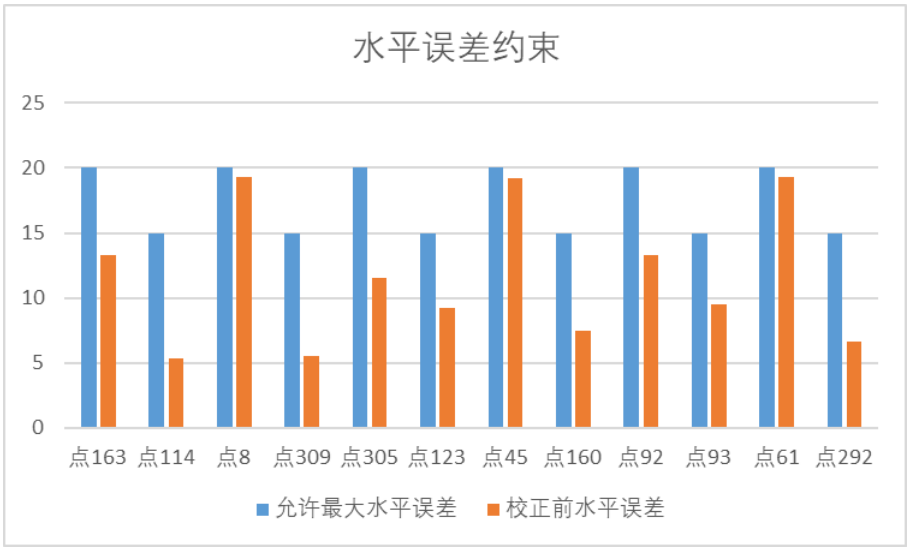



图 22 问题二数据集 2 各校正点前水平误差约束检查

### 4.2.4 考虑最小转弯半径的航迹优选校正点的选择

问题二中，因采用 Dubins 曲线且校准点数目繁多，为减少在计算两点之间的 Dubins 距离的耗时，以达到题目要求的快速规划。我们以问题一为基础，做起点 A 与终点 B 的连线 AB，计算问题一结果中，航迹中各校正点 P 与连线 AB 的距离 d。根据 d 合理设定优选校准参数 a，判断该校正点是否属于优选校准点。

为实现航迹的快速规划，优先考虑优选校准点作为航迹点进行规划。在求解过程中，对优选校准参数及优选校准点进行了灵敏度分析，实验证明，优选校准点可以准确、快速的进行航迹的规划与预处理。

根据问题一的两个规划结果，可以求得途中各校正点 P 与 AB 点直航航迹的直线距离

$$d = \frac{||\overrightarrow{PA} \otimes \overrightarrow{BA}||}{||\overrightarrow{BA}||} \tag{29}$$

从表 11、表 12 问题二数据集 2 航迹点与 AB 直航航迹距离可以看出，数据集 1 的规划结果中，最终航迹各校正点与 AB 直航航迹的距离都小于 6000m，数据集 2 的规划结果中，最终航迹各校正点与 AB 直航航迹的距离都小于 4500m。

为实现题目要求的快速规划，在求解问题 2 时可缩小搜索空间规模，预先排除距离直航航迹较远的校正点，以提升求解效率。

表 11 问题二数据集 1 航迹点与 AB 直航航迹距离

| 校正点编号 | 校正点与 AB 连线的距离 |
|---|---|
| **0** | 0 |
| **503** | 5915.76 |
| **200** | 5686.16 |
| **80** | 4838.43 |
| **237** | 5478.53 |
| **170** | 3417.31 |
| **278** | 1361.45 |
| **369** | 2551.72 |
| **214** | 1335.14 |
| **397** | 530.41 |
| **612** | 0 |

表 12 问题二数据集 2 航迹点与 AB 直航航迹距离

| 校正点编号 | 校正点与 AB 连线的距离 |
|---|---|
| **0** | 0 |
| **163** | 706.33 |
| **114** | 2096.90 |
| **8** | 4435.65 |
| **309** | 1410.33 |
| **305** | 1995.52 |
| **123** | 160.90 |
| **45** | 2002.01 |

| | |
|---|---|
| **160** | 1833.58 |
| **92** | 941.87 |
| **93** | 1351.36 |
| **61** | 506.63 |
| **292** | 3033.42 |
| **326** | 0 |

表 13 各数据集中距离 AB 直航航迹小于特定距离的校正点数目

| 距 AB 直航航迹的距离（m） | 数据集 1 | 数据集 2 |
|---|---|---|
| **<8000** | 60 | 59 |
| **<10000** | 103 | 72 |
| **<12000** | 136 | 92 |
| **<20000** | 250 | 151 |
| **无限制** | 611 | 325 |

可见，经过计算与直航航迹的距离，可以快速的缩小备选校正点的数目，在计算三维 Dubis 曲线时，可优先计算优选航迹点的 Dubins 距离矩阵，以减小计算整个数据集的 Dubins 距离矩阵花费的时间。

## 4. 3 问题三建模与求解

### 4.3.1 带有校正概率的航迹快速规划模型

问题三中，为沿用问题一的模型进行求解。在概率校正点 i 经过概率校正后，若校正成功，则剩余误差为 0；若校正失败，则剩余误差为$\min(\Delta x_i, 5)$。

在此，我们首先考虑概率校正点最恶劣的情况，即不论校正成功与否，均将概率校正点的校正水平降到最低，即校正成功概率为 0，概率校正点 i 校正后剩余误差统一记为 $\min(\Delta x_i,5)$。那么，问题一模型的公式（9）、公式（10）

$$\mathrm{M}\left(1-x_{ij}\right)+\Delta x_j \geq \delta d_{ij}+\Delta x_i \cdot l_i, \forall i, j \sum_{i \in C} \sum_{j=1}^{n} x_{ij} \tag{9}$$

$$\mathrm{M}\left(1-x_{ij}\right)+\Delta y_j \geq \delta d_{ij}+\Delta y_i \cdot (1-l_i), \forall i, j \sum_{i \in C} \sum_{j=1}^{n} x_{ij} \tag{10}$$

将变更为

$$\mathrm{M}\left(1-x_{ij}\right)+\Delta x_j \geq \delta d_{ij}+\Delta x_i \cdot l_i+(1-l_i) w_i \cdot \min(\Delta x_i, 5), \forall i, j \sum_{i \in C} \sum_{j=1}^{n} x_{ij} \tag{31}$$

$$\mathrm{M}\left(1-x_{ij}\right)+\Delta y_j \geq \delta d_{ij}+\Delta y_i \cdot (1-l_i)+l_i w_i \cdot min(\Delta y_i, 5), \forall i, j \sum_{i \in C} \sum_{j=1}^{n} x_{ij} \tag{32}$$

其中，$w_i$为校正点属性，若校正点 i 为概率校正点，则$w_i$为 1，校正后仍有 5 个单位的误差。若校正点 i 为普通校正点，则$w_i$为 0。校正后剩余误差为 0。

### 4.3.2 带有校正概率的航迹快速规划模型求解结果

数据集 1 可以由 CPLEX 求得最优解。经过的校正点顺序依次为 0，503，69，506，371，183，194，450，113，485，248，612，总距离为 104827.37m。数据集 1 航迹共经过 12 个点，除起点与终点外，经过 10 个航迹校正点，其中 5 个水平校正点（1 个概率校

正点）。5 个垂直校正点（2 个概率校正点）。

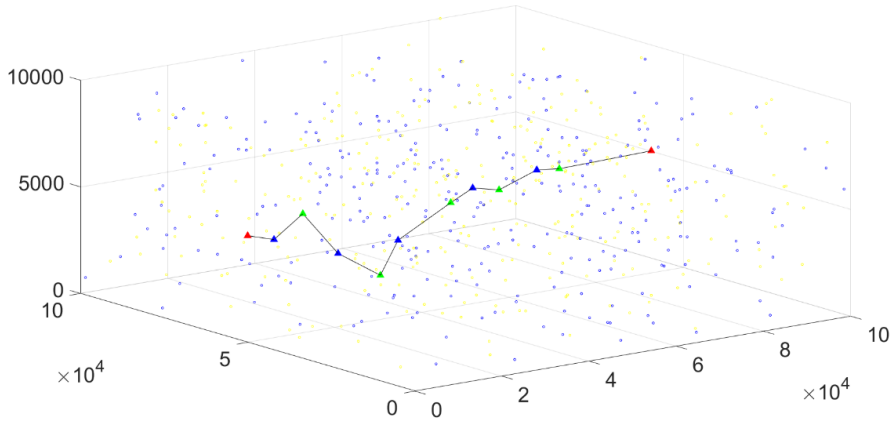



图 23 问题三数据集 1 航迹图

表 14 问题三数据集 1 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **503** | 13.39 | 13.39 | 12 |
| **69** | 13.81 | 22.20 | 02 |
| **506** | 21.68 | 12.87 | 12 |
| **371** | 15.62 | 23.48 | 01 |
| **183** | 22.65 | 7.04 | 11 |
| **194** | 13.61 | 20.65 | 01 |
| **450** | 19.59 | 5.98 | 11 |
| **113** | 6.51 | 12.48 | 01 |
| **485** | 13.79 | 7.28 | 11 |
| **248** | 4.22 | 11.50 | 01 |
| **612** | 23.73 | 19.51 | 终点 B |

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正点允许的最大垂直误差与水平误差（见图 24、图 25），抵达终点的误差范围也不超过 $\theta$。

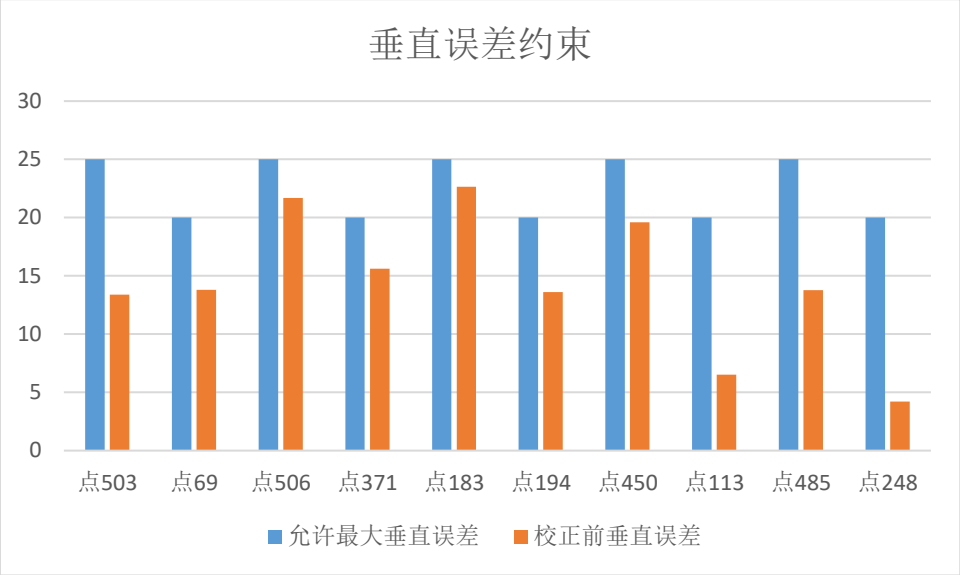



图 24 问题三数据集 1 各校正点前垂直误差约束检查

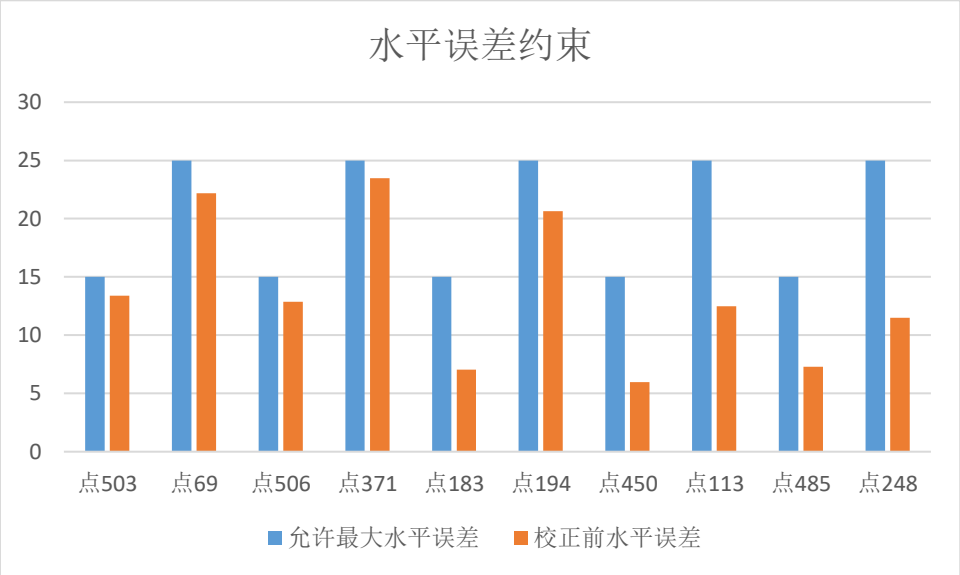



图 25 问题三数据集 1 各校正点前水平误差约束检查

数据集 2 可以由 CPLEX 求得最优解。经过的校正点顺序依次为 0，169，322，270，89，236，132，53，112，268，250，243。73，249，274，12，216，16，282，141，291，161，326，总距离为 161650.41m。数据集 1 航迹共经过 23 个点，除起点与终点外，经过 21 个航迹校正点，其中 11 个水平校正点（1 个概率校正点）。10 个垂直校正点。

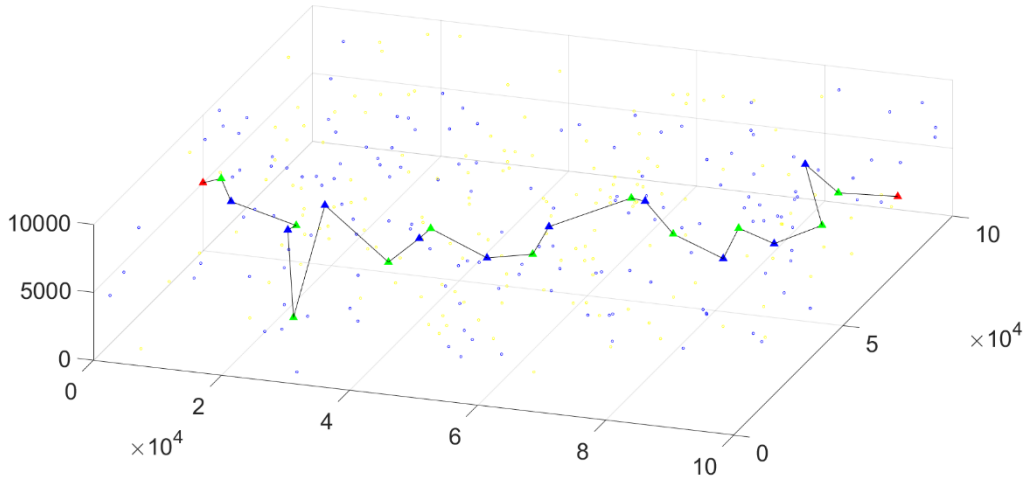



图 26 问题三数据集 2 航迹图

表 15 问题三数据集 2 航迹规划结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 | 校正点类型 |
|---|---|---|---|
| **0** | 0 | 0 | 出发点 A |
| **169** | 9.27 | 9.27 | 01 |
| **322** | 13.42 | 4.15 | 11 |
| **270** | 11.34 | 15.49 | 01 |
| **89** | 18.89 | 7.55 | 11 |
| **236** | 10.13 | 17.68 | 01 |
| **132** | 19.83 | 9.70 | 11 |
| **53** | 10.26 | 19.96 | 01 |
| **112** | 15.46 | 5.20 | 11 |
| **268** | 2.16 | 7.36 | 01 |
| **250** | 11.76 | 9.61 | 11 |
| **243** | 6.96 | 16.57 | 01 |
| **73** | 10.50 | 3.54 | 11 |
| **249** | 12.85 | 16.39 | 01 |
| **274** | 15.69 | 2.84 | 11 |
| **12** | 6.44 | 9.28 | 01 |
| **216** | 14.24 | 7.80 | 11 |
| **16** | 4.22 | 12.02 | 01 |
| **282** | 11.66 | 7.44 | 11 |
| **141** | 8.10 | 15.54 | 01 |
| **291** | 13.59 | 5.49 | 11 |
| **161** | 6.47 | 11.95 | 02 |
| **326** | 16.61 | 15.15 | 终点 B |

经检查，经过各航迹校正点前飞行器的垂直误差与水平误差，均不超过该航迹校正

点允许的最大垂直误差与水平误差（见图 27、图 28），抵达终点的误差范围也不超过 $\theta$。

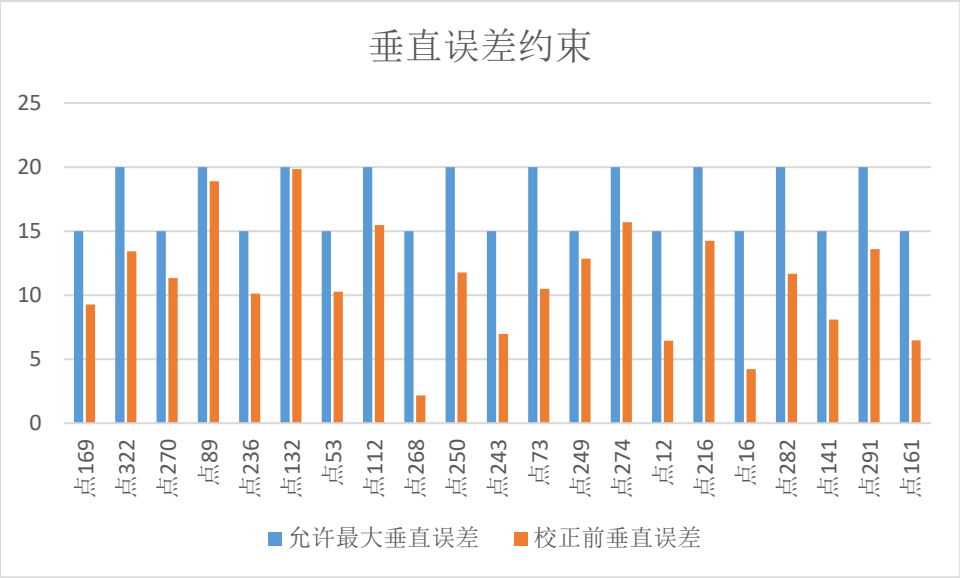



图 27 问题三数据集 2 各校正点前垂直误差约束检查

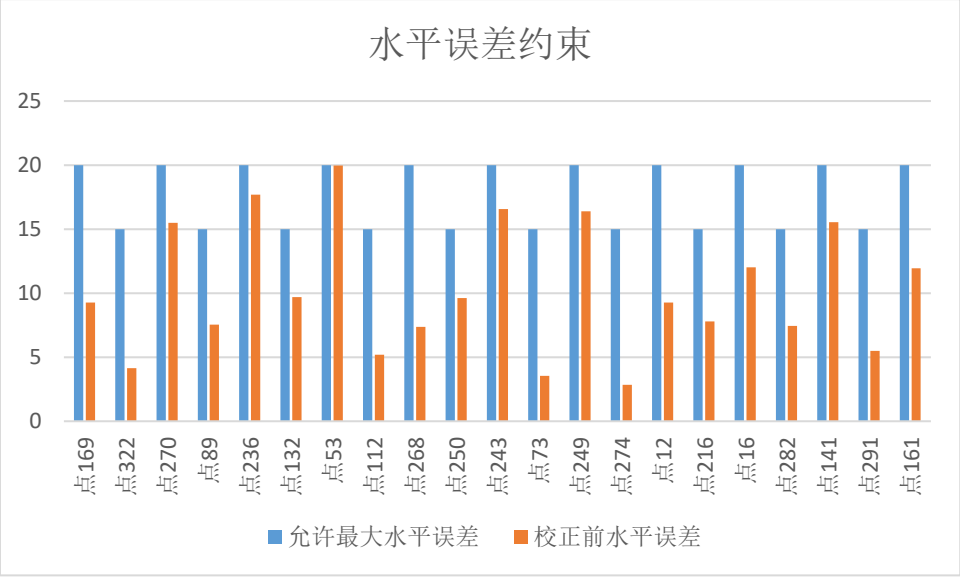



图 28 问题三数据集 2 各校正点前水平误差约束检查

### 4.3.3 带有校正概率的航迹快速规划蚁群算法

因问题三中，两个数据集具有特殊性，在概率校正点最差的情况下仍能生成 100%概率到达终点 B 的航迹。为解决更普遍情况下带有校正概率的航迹规划问题，我们提出了带校正概率的快速规划模型，并采用蚁群算法进行求解。

若当前航迹 $R(R = \{r_A \dots r_i \dots r_B\})$共有 $n$ 个点，其中有 $m(m \leq n-2)$个随机校正点。因随机校正点存在校正概率 $p(p=0.8)$，经过每个随机校正点后，校正误差可能为 0，或仍

有剩余误差$min(error,5)$。因此路径最终存在 $2^m$ 种校正结果，产生每种校正结果的概率为$1/2^m$。

$S_i$ 表示每种校正结果生成的航迹能否到达终点，若能到达 B 点，则 $S_i$ 为 1，若无法到达 B 点，则 $S_i$ 为 0。

$P_R$ 表示航迹 $R$ 最终到达 B 点的概率

$$P_R = \frac{\sum_{i=1}^{2^m} S_i}{2^m} \tag{33}$$

蚁群算法(ACO)是一种模拟蚂蚁觅食行为的模拟优化算法，它是由意大利学者 Dorigo M[8]等人于 1991 年首先提出，并首先使用在解决 TSP（旅行商问题）上。蚁群算法的基本原理为:

1、蚂蚁在路径上释放信息素。

2、碰到还没走过的路口，就随机挑选一条路。同时，释放与路径长度有关的信息素。

3、信息素浓度与路径长度成反比。后来的蚂蚁再次碰到该路口时，就选择信息素浓度较高路径。

4、最优路径上的信息素浓度越来越大。

5、最终蚁群找到最优寻食路径。

求解中，假设蚁群算法中的每只蚂蚁是具有以下特征的简单智能体：每次周游，每只蚂蚁在其经过的支路（i,j）上都留下信息素。蚂蚁选择校准点的概率与校准点之间的距离和当前连接支路上所包含的信息素余量有关。

为了强制蚂蚁进行合法的周游，直到一次周游完成后，才允许蚂蚁游走已访问过的校准点。

蚁群算法的基本实现步骤如下：

| **快速规划航迹蚁群算法** | |
|---|---|
| 步骤 1. | 参数初始化，设置最大循环次数，将 $m$ 只蚂蚁，放在 $n$ 个校正点上，令有向图上每条边（$i,j$）的初始化信息量$\Delta\tau_{ij}(t)$为常数，初始时刻$\Delta\tau_{ij}(0)=0$ |
| 步骤 2. | 循环次数+1 |
| 步骤 3. | 蚂蚁数目 $k$=$k$+1 |
| 步骤 4. | 如果到了算法规定的 $H$ 时刻，则信息素会挥发 |
| 步骤 5. | 根据蚂蚁当前状态，计算下一步可行的点的集合 S。若蚂蚁未走至终点且已无可行点，则清空路径和禁忌表，返回步骤 4。 |
| 步骤 6. | 蚂蚁个体根据状态转移概率式，计算选择概率，从 S 中选择校正点 $j$ 并前进。 |
| 步骤 7. | 把蚂蚁移动至选择的新校准点，并将并将该校准点加入蚂蚁的禁忌表中 |
| 步骤 8. | 若蚂蚁未走至终点，该蚂蚁的禁忌表清空，则返回步骤 4 |
| 步骤 9. | 依据公式更新信息素 |
| 步骤 10. | 若满足结束条件，即循环次数大于最大循环次数，则结束算法输出结果。否则，跳转至步骤 2 |

使用概率转移准则表示蚂蚁 $k$ 在 $t$ 时刻由元素 i 转移到元素 $j$ 的转移概率。即，

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ik}(t)]^{\beta}}{\sum_{N\subset allowed_k}[\tau_{ir}(t)]^{\alpha}[\eta_{ir}(t)]^{\beta}}, & j\in allowed_k, s\subset allowed_k \\ 0, \text{否则} & \end{cases}$$

其中，$allowed_k$ 表示蚂蚁下一步允许选择的城市；α 为信息启发式因子，$\tau_{ij}$ 为校正点间的路径信息素。该启发因子作为此路径信息素的幂数，代表了该条路径的相对重要性，α 越大，蚂蚁选择该条路径的可能性就越大。但如果 α 过大，则蚂蚁会过分依赖前蚂蚁的经验，其选择的随机性减少，容易陷入局部最优。若 α 过小，则算法不能充分以前蚂蚁的经验指导现在蚂蚁的下一部动作，容易造成收敛过慢。B 表示能见度的相对重要性。$\eta_{ij}(t)$ 为启发函数，表示了蚂蚁从校正点 i 转移到校正点 j 的期望成都，反映了寻优过程中确定性因素的作用强度，值越大，则蚂蚁在城市 i 选择局部最短路径的可能性越大，此时算法会很快收敛，会影响蚂蚁搜索的随机性，容易陷入局部最优。

### 4.3.4 带有校正概率的航迹快速规划蚁群算法求解结果

针对数据集 1，设置蚁群算法运行时间为一分钟，得到的最好结果解为 0，303，199，82，534，316，453，11，403，113，448，397，612。到达终点的概率为 100%，距离为 133234.86，到达终点的概率相同，航迹总距离劣于 4.3.2 中带校正概率的快速规划模型求得的解。

表 16 问题三数据集 1 蚁群算法结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 |
|---|---|---|
| **0** | 0 | 0 |
| **303** | 17.6742 | 17.6742 |
| **199** | 24.9608 | 7.2866 |
| **82** | 5.0581 | 12.3447 |
| **534** | 18.5721 | 13.514 |
| **316** | 10.8138 | 24.3278 |
| **453** | 23.1948 | 12.381 |
| **11** | 11.9391 | 24.3201 |
| **403** | 24.78 | 12.8409 |
| **113** | 7.5293 | 20.3702 |
| **448** | 11.8145 | 4.2853 |
| **397** | 24.7545 | 12.9399 |
| **612** | 16.9727 | 29.9126 |

针对数据集 2，设置蚁群算法运行时间为一分钟，运行时间内得到的最好解为 0，169，266，100，270，89，236，132，53，112，143，250，86，167，249，274，16，186，279，282，141，291，61，292，326。到达终点的概率为 80%，距离为 197388.18m，到达终点的概率和距离均劣于 4.3.2 中带校正概率的快速规划模型所得解。

表 17 问题三数据集 2 蚁群算法结果表

| 校正点编号 | 校正前垂直误差 | 校正前水平误差 |
|---|---|---|
| **0** | 0 | 0 |
| **169** | 9.2705 | 9.2705 |
| **266** | 18.7485 | 9.478 |
| **100** | 7.1399 | 16.6179 |
| **270** | 9.7475 | 2.6076 |
| **89** | 17.2982 | 7.5508 |
| **236** | 10.1274 | 17.6782 |
| **132** | 19.8316 | 9.7042 |
| **53** | 10.2592 | 19.9634 |
| **112** | 15.4621 | 5.2029 |
| **143** | 13.2032 | 18.4061 |
| **250** | 19.5478 | 6.3446 |
| **86** | 10.6901 | 17.0347 |
| **167** | 14.4589 | 3.7688 |
| **249** | 13.8845 | 17.6533 |
| **274** | 16.7252 | 2.8407 |
| **16** | 14.3452 | 17.1859 |
| **186** | 18.9822 | 4.637 |
| **279** | 11.199 | 15.836 |
| **282** | 18.579 | 7.38 |
| **141** | 8.1003 | 15.4804 |
| **291** | 13.5863 | 5.4859 |
| **61** | 10.654 | 16.1399 |
| **292** | 17.2079 | 6.5539 |
| **326** | 6.9605 | 13.5144 |

## 五、总结

本文主要通过建立整数规划模型、运用几何学方法及飞行性能相结合的设计思路，提出了智能飞行器航路快速规划算法。

在求解问题一时，本文在考虑最短航迹距离的基础上，考虑最小化经过的校正点数目。建立混合整数规划模型，设定了优化目标的优先级，使之成为单目标优化问题，调用 Cplex 可在秒级时间内求得了模型的最优解。

求解问题二时，在问题一的航迹规划基础上，引入 Dubins 曲线，精确解得因转弯造成的航迹变化。Dubins 曲线的入射速度方向与出射速度方向均为决策变量，为简化计算量，提升求解效率，本文假设速度方向均朝向终点方向。在之后的研究中可以加强对速度方向的研究。在求得 Dubins 距离矩阵后，通过 Cplex 求解，可以在秒级时间内求得问题的最优解。为了满足飞行器航迹规划的实时决策，为进一步提升求解时间，本文提出了优选校正点概念，在不影响解的优越性的情况下极大缩小 Dubins 矩阵规模，提升求解效率。

针对问题三，通过收紧概率约束，将机会约束问题简化，使之可以通过整数规划求

得最优解。通过生成航迹，可计算成功到达终点的概率。为了提升求解方法的通用性，本文引进了具有全局搜索性能的蚁群算法，可在一定时间内求得较优解。然而，蚁群算法具有搜索时间长、算法结果对参数敏感、对大空间问题收敛速度较慢的特性，若问题规模进一步增大，仍需进一步结合问题进行算法的改进，以提升蚁群算法的求解效率，为智能飞行器的航迹快速规划提供实时决策。

# 六、参考文献


[1] 董世建. 复杂约束条件下航迹规划方法研究[D]. 2016.

[2] 不确定环境下无人机航迹动态规划及仿真研究[D]. 南京航空航天大学, 2013.

[3] L.E. Dubins, “On curves of minimal length with a constraint onaverage curvature, and with prescribed initial and terminal positions and tangents”, American Journal of Mathematics, 79 (1957) 497-516.

[4] Hota S , Ghose D . Optimal geometrical path in 3D with curvature constraint[C]// IEEE/RSJ International Conference on Intelligent Robots & Systems. IEEE, 2010.

[5] 陈海, 王新民, 焦裕松,等. 无人机覆盖路径规划中转弯机动的运动学分析[J]. 飞行力学, 2010, 28(2):31-34.

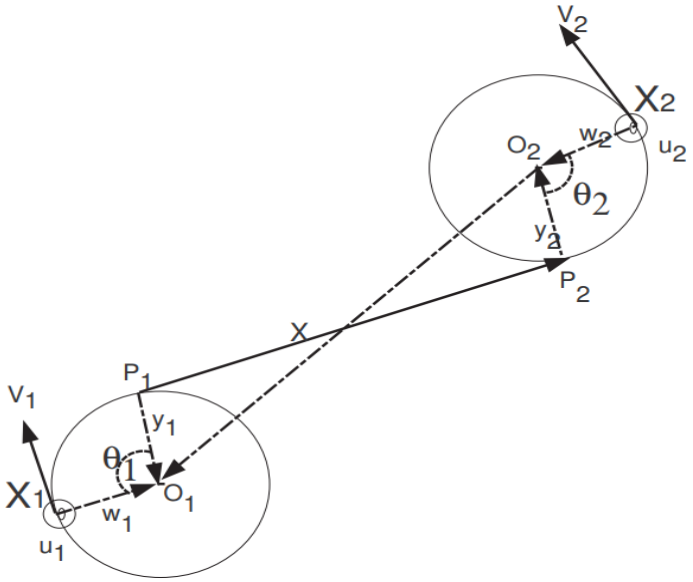
[6] 武中, 邹鹏. 含机会约束的两阶段电动汽车充电基础设施随机规划[J]. 智能电网, 2017(4).

[7] 屈高强, 李荣, 董晓晶, et al. 基于随机机会约束规划的有源配电网多目标规划[J]. 电力建设, 2015, 36(11):10-16.

[8] Dorigo M, Gambardella L M. Ant colony system: A cooperative learning approach to the traveling salesman problem[J]. IEEE Trans on Ec, 1997, 1(1):53-66.

# 七、附录

## 7.1 Dubins 曲线推导过程




如上图所示，按照最小曲率半径转弯的初始曲线和最终曲线处于不同的平面，两曲线之间的直线部分是两个平面的交线，记这两个平面的公共向量都是$X(X_x,X_y,X_z)$，则单位向量为：

$$x=\frac{X}{\|X\|} \tag{8.1}$$

给定的初始位置的单位速度向量：

$$v_1=\frac{V_1}{\|V_1\|} \tag{8.2}$$

则垂直于初始曲线圆平面的法向量为：

$$U_1=X\times V_1 \tag{8.3}$$

得到单位法向量：

$$u_1=\frac{U_1}{\|U_1\|} \tag{8.4}$$

得到从初始位置朝向第一个圆中心的半径向量是：

$$W_1=V_1\times U_1=V_1\times(X\times V_1) \tag{8.5}$$

单位化：

$$w_1=\frac{W_1}{\|W_1\|} \tag{8.6}$$

则第一个圆的圆心为：

$$o_1 = X_1 + rw_1 \tag{8.7}$$

两圆之间直线与第一个圆的切点到圆心的半径向量为：

$$Y_1 = X \times U_1 = X \times (X \times V_1) \tag{8.8}$$

单位化：

$$y_1 = \frac{Y_1}{\| Y_1 \|} \tag{8.9}$$

从而得到该切点为：

$$P_1 = o_1 - ry_1 = X_1 + rw_1 - ry_1 \tag{8.10}$$

同理，设给定的终点位置的单位速度向量：

$$v_2 = \frac{V_2}{\| V_2 \|} \tag{8.11}$$

则垂直于第二个曲线圆平面的法向量为：

$$U_2 = X \times V_2 \tag{8.12}$$

得到单位法向量：

$$u_2 = \frac{U_2}{\| U_2 \|} \tag{8.13}$$

得到从终点位置朝向第二个圆中心的半径向量是：

$$W_2 = -V_2 \times U_2 = V_2 \times (X \times V_2) \tag{8.14}$$

单位化：

$$w_2 = \frac{W_2}{\| W_2 \|} \tag{8.15}$$

则第二个圆的圆心为：

$$o_2 = X_2 + rw_2 \tag{8.16}$$

两圆之间直线与第二个圆的切点到圆心的半径向量为：

$$Y_2 = -X \times U_2 = -X \times (X \times V_2) \tag{8.17}$$

单位化：

$$y_2 = \frac{Y_2}{\| Y_2 \|} \tag{8.18}$$

从而得到该切点为：

$$P_2 = o_2 - ry_2 = X_2 + rw_2 - ry_2 \tag{8.19}$$

$X$ 可由以下公式解出

$$X = P_2 - P_1 = (X_2 + rw_2 - ry_2) - (X_1 + rw_1 - ry_1) \tag{8.20}$$

引入两个角度$\theta_1$和$\theta_2$，分别代表两个圆上的转向角，则 $X$ 可表示为：

$$X = X_2 - X_1 - r(x + v_1)\tan\frac{\theta_1}{2} - r(x + v_2)\tan\frac{\theta_2}{2} \tag{8.21}$$

进一步简化得到以下结果：

$$cos\theta_1 = v_1 \cdot x \tag{8.22}$$

$$cos\theta_2 = v_2 \cdot x \tag{8.23}$$

$$X_{2x} - X_{1x} = X_x + rx_x[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2x}\tan\frac{\theta_2}{2} + v_{1x}\tan\frac{\theta_1}{2}] \tag{8.24}$$

$$X_{2y} - X_{1y} = X_y + rx_y[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2y}\tan\frac{\theta_2}{2} + v_{1y}\tan\frac{\theta_1}{2}] \tag{8.25}$$

$$X_{2z} - X_{1z} = X_z + rx_z[\tan\frac{\theta_1}{2} + \tan\frac{\theta_2}{2}] + r[v_{2z}\tan\frac{\theta_2}{2} + v_{1y}\tan\frac{\theta_1}{2}] \tag{8.26}$$

最终的 Dubins 曲线由三段组成，第一段弧长、直线长度、第二段弧长计算公式分别为(8.27)~(8.29)

$$rcos^{-1}(w_1 \cdot y_1) \tag{8.27}$$

$$(X_2 + rw_2 - ry_2) - (X_1 + rw_1 - ry_1) \tag{8.28}$$

$$rcos^{-1}(w_2 \cdot y_2) \tag{8.29}$$

经由上述方程，可以解得四种类型的 CSC 路径，下面给出了完整的方程组

$$X = (X_2 - X_1) \mp r(x + v_1)\frac{tan\theta_1}{2} \mp r(x + v_2)\frac{tan\theta_2}{2} \tag{8.30}$$

因四类 CSC 路径均为可行路径，因此在应用中，我们选择其中最短的路径。

## 7.2 算法实现代码

### 7.2.1 问题一

```
// 初始化
cplex = new IloCplex();
x = new IloNumVar[pointsNum][];
level = new IloNumVar[pointsNum];
vert = new IloNumVar[pointsNum];
```

```
for(int i = 0; i < pointsNum; i++) {
    x[i] = cplex.numVarArray(pointsNum, 0, 1, IloNumVarType.Int);
    level[i] = cplex.numVar(0, M1, IloNumVarType.Float);
    vert[i] = cplex.numVar(0, M1, IloNumVarType.Float);
}

// 优化目标：最小距离
IloNumExpr obj1 = cplex.numExpr();
IloNumExpr obj2 = cplex.numExpr();
for(int i = 0; i < pointsNum; i++) {
    for(int j = 0; j < pointsNum; j++) {
        obj1 = cplex.sum(obj1, cplex.prod(d[i][j], x[i][j]));
    }
}
for(int i = 0; i < pointsNum; i++) {
    for(int j = 1; j < pointsNum - 1; j++) {
        obj2 = cplex.sum(obj2, x[i][j]);
    }
}
cplex.addMinimize(obj1);

// 约束 1：每个节点最多访问一次,且不访问自己
for(int i = 0; i < pointsNum; i++) {
    cplex.addLe(cplex.sum(x[i]), 1);
    cplex.addEq(x[i][i], 0);
}
// 约束 2：流平衡
for(int i = 1; i < pointsNum - 1; i++) {
    IloNumExpr expr = cplex.numExpr();
    for(int j = 0; j < pointsNum; j++) {
        expr = cplex.sum(expr, x[j][i]);
    }
    cplex.addEq(cplex.sum(x[i]), expr);
}
// 约束 3：必须从 A 点出发
cplex.addEq(cplex.sum(x[0]), 1);
// 约束 4：必须回到 B 点
IloNumExpr expr = cplex.numExpr();
for(int i = 0; i < pointsNum - 1; i++) {
    expr = cplex.sum(expr, x[i][pointsNum - 1]);
}
cplex.addEq(expr, 1);
cplex.addEq(cplex.sum(x[pointsNum - 1]), 0);
// 约束 5：水平和垂直误差
```

```
for(int j = 1; j < pointsNum; j++) {
    for(int i = 0; i < pointsNum - 1; i++) {
        cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
level[j]), cplex.sum(t*d[i][j], cplex.prod(level[i], points.get(i).label)));
        cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
vert[j]), cplex.sum(t*d[i][j], cplex.prod(vert[i], (1 -
points.get(i).label))));
    }
} // 误差传递
for(int i = 0; i < pointsNum - 1; i++) {
    // 垂直误差校正点
    if(points.get(i).label == 1) {
        cplex.addLe(level[i], a2);
        cplex.addLe(vert[i], a1);
    }
    else {
        cplex.addLe(level[i], b2);
        cplex.addLe(vert[i], b1);
    }
}
// 约束 6：终点误差
cplex.addLe(level[pointsNum - 1], c);
cplex.addLe(vert[pointsNum - 1], c);
```

### 7.2.2 问题二

求解 3DDubins 曲线

（1）第一种情况

```
function F = myfunTest1(X)
    global v1 A v2 B r;
    F(1) = cos(X(4)) -
(v1(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(2) = cos(X(5)) -
(v2(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(3) = B(1) - A(1) - X(1) -
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(1))*tan(X(4)/2) -
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(1))*tan(X(5)/2);
    F(4) = B(2) - A(2) - X(2) -
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(2))*tan(X(4)/2) -
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(2))*tan(X(5)/2);
```

```
    F(4) = B(3) - A(3) - X(3) -
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(3))*tan(X(4)/2) -
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(3))*tan(X(5)/2);
end
```

（2）第二种情况

```
function F = myfunTest2(X)
    global v1 A v2 B r;
    F(1) = cos(X(4)) -
(v1(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(2) = cos(X(5)) -
(v2(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(3) = B(1) - A(1) - X(1) +
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(1))*tan(X(4)/2) -
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(1))*tan(X(5)/2);
    F(4) = B(2) - A(2) - X(2) +
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(2))*tan(X(4)/2) -
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(2))*tan(X(5)/2);
    F(4) = B(3) - A(3) - X(3) +
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(3))*tan(X(4)/2) -
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(3))*tan(X(5)/2);
end
```

（3）第三种情况

```
function F = myfunTest3(X)
    global v1 A v2 B r;
    F(1) = cos(X(4)) -
(v1(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(2) = cos(X(5)) -
(v2(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(3) = B(1) - A(1) - X(1) +
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(1))*tan(X(4)/2) +
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(1))*tan(X(5)/2);
    F(4) = B(2) - A(2) - X(2) +
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(2))*tan(X(4)/2) +
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(2))*tan(X(5)/2);
    F(4) = B(3) - A(3) - X(3) +
```

```
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(3))*tan(X(4)/2) +
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(3))*tan(X(5)/2);
end
```

（4）第四种情况

```
function F = myfunTest4(X)
    global v1 A v2 B r;
    F(1) = cos(X(4)) -
(v1(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v1(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(2) = cos(X(5)) -
(v2(1)*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(2)*(X(2)/sq
rt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3)))+v2(3)*(X(3)/sqrt(X(1)*X(1)+X(
2)*X(2)+X(3)*X(3))));
    F(3) = B(1) - A(1) - X(1) -
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(1))*tan(X(4)/2) +
r*(X(1)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(1))*tan(X(5)/2);
    F(4) = B(2) - A(2) - X(2) -
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(2))*tan(X(4)/2) +
r*(X(2)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(2))*tan(X(5)/2);
    F(4) = B(3) - A(3) - X(3) -
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v1(3))*tan(X(4)/2) +
r*(X(3)/sqrt(X(1)*X(1)+X(2)*X(2)+X(3)*X(3))+v2(3))*tan(X(5)/2);
end
```

（5）计算距离矩阵

```
clear;
clc;
global v1 A v2 B r;
r = 200;
data1 = importdata('data1.txt');
pointNum = size(data1, 1);
distance = [];

distanceU = [];
for i = 1 : pointNum
    M = data1(i, 2:4);
    for j = 1 : pointNum
        if i==j
            distanceU(i, j) = 0;
        end
        N = data1(j, 2:4);
        L = M-N;
        distanceU(i, j) = sqrt(sum(L.*L));
    end
```

```
end

endPoint = data1(pointNum, 2:4);
for i = 1 : 1
    for j = 504 : 504
        if distanceU(i,j) > 25000 && j < pointNum
            distance(i,j) = distanceU(i,j);
            continue;
        end
        if i==j
            distance(i,i)=0;
        else
            minDistance = 9999999999;
            A = data1(i, 2:4);
            B = data1(j, 2:4);
            V1 = endPoint - A;
            V2 = endPoint - B;
            if j==pointNum
                V2 = B - A;
            end
            v1 = V1./sqrt(sum(V1.*V1));
            v2 = V2./sqrt(sum(V2.*V2));
            options = optimset('TolFun',1e-10);
            K = [2,2,2,2,2];

            D = fsolve(@myfunTest1, K, options);
            X = D(1:3);
            x = X./sqrt(sum(X.*X));
            U1 = cross(X,V1);
            u1 = U1./sqrt(sum(U1.*U1));
            W1 = cross(V1, U1);
            w1 = W1./sqrt(sum(W1.*W1));
            o1 = A + r*w1;
            Y1 = cross(X, U1);
            y1 = Y1./sqrt(sum(Y1.*Y1));
            P1 = o1 - r*y1;
            U2 = cross(X, V2);
            u2 = U2./sqrt(sum(U2.*U2));
            W2 = -1 * cross(V2, U2);
            w2 = W2./(sqrt(sum(W2.*W2)));
            o2 = B + r*w2;
            Y2 = -1 * cross(X, U2);
            y2 = Y2./sqrt(sum(Y2.*Y2));
            P2 = o2 - r*y2;
```

```
arc1 = r*acos(sum(w1.*y1));
line = sqrt(sum((P2-P1).*(P2-P1)));
arc2 = r*acos(sum(w2.*y2));
totalDistance = arc1 + line + arc2;
minDistance = min(minDistance, totalDistance);


D = fsolve(@myfunTest2, K, options);
X = D(1:3);
x = X./sqrt(sum(X.*X));
U1 = cross(X,V1);
u1 = U1./sqrt(sum(U1.*U1));
W1 = cross(V1, U1);
w1 = W1./sqrt(sum(W1.*W1));
o1 = A + r*w1;
Y1 = cross(X, U1);
y1 = Y1./sqrt(sum(Y1.*Y1));
P1 = o1 - r*y1;
U2 = cross(X, V2);
u2 = U2./sqrt(sum(U2.*U2));
W2 = -1 * cross(V2, U2);
w2 = W2./(sqrt(sum(W2.*W2)));
o2 = B + r*w2;
Y2 = -1 * cross(X, U2);
y2 = Y2./sqrt(sum(Y2.*Y2));
P2 = o2 - r*y2;

arc1 = r*acos(sum(w1.*y1));
line = sqrt(sum((P2-P1).*(P2-P1)));
arc2 = r*acos(sum(w2.*y2));
totalDistance = arc1 + line + arc2;
minDistance = min(minDistance, totalDistance);

D = fsolve(@myfunTest3, K, options);
X = D(1:3);
x = X./sqrt(sum(X.*X));
U1 = cross(X,V1);
u1 = U1./sqrt(sum(U1.*U1));
W1 = cross(V1, U1);
w1 = W1./sqrt(sum(W1.*W1));
o1 = A + r*w1;
Y1 = cross(X, U1);
y1 = Y1./sqrt(sum(Y1.*Y1));
```

```
P1 = o1 - r*y1;
U2 = cross(X, V2);
u2 = U2./sqrt(sum(U2.*U2));
W2 = -1 * cross(V2, U2);
w2 = W2./(sqrt(sum(W2.*W2)));
o2 = B + r*w2;
Y2 = -1 * cross(X, U2);
y2 = Y2./sqrt(sum(Y2.*Y2));
P2 = o2 - r*y2;

arc1 = r*acos(sum(w1.*y1));
line = sqrt(sum((P2-P1).*(P2-P1)));
arc2 = r*acos(sum(w2.*y2));
totalDistance = arc1 + line + arc2;
minDistance = min(minDistance, totalDistance);


D = fsolve(@myfunTest4, K, options);
X = D(1:3);
x = X./sqrt(sum(X.*X));
U1 = cross(X,V1);
u1 = U1./sqrt(sum(U1.*U1));
W1 = cross(V1, U1);
w1 = W1./sqrt(sum(W1.*W1));
o1 = A + r*w1;
Y1 = cross(X, U1);
y1 = Y1./sqrt(sum(Y1.*Y1));
P1 = o1 - r*y1;
U2 = cross(X, V2);
u2 = U2./sqrt(sum(U2.*U2));
W2 = -1 * cross(V2, U2);
w2 = W2./(sqrt(sum(W2.*W2)));
o2 = B + r*w2;
Y2 = -1 * cross(X, U2);
y2 = Y2./sqrt(sum(Y2.*Y2));
P2 = o2 - r*y2;

arc1 = r*acos(sum(w1.*y1));
line = sqrt(sum((P2-P1).*(P2-P1)));
arc2 = r*acos(sum(w2.*y2));
totalDistance = arc1 + line + arc2;
minDistance = min(minDistance, totalDistance);

distance(i,j) = minDistance;
```

```
        end
    end
end
```

### 7.2.3 紧约束下的问题三求解代码：

```
// 约束 5：水平和垂直误差
for(int j = 1; j < pointsNum; j++) {
    for(int i = 0; i < pointsNum - 1; i++) {
        if(points.get(i).success == 1) {
            if(points.get(i).label == 1) {
                cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
level[j]), cplex.sum(t*d[i][j], level[i]));
                cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
vert[j]), cplex.sum(t*d[i][j], cplex.min(vert[i], 5)));
            }
            else {
                cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
level[j]), cplex.sum(t*d[i][j], cplex.min(level[i], 5)));
                cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
vert[j]), cplex.sum(t*d[i][j], vert[i]));
            }
        }
        else{
            cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
level[j]), cplex.sum(t*d[i][j], cplex.prod(level[i], points.get(i).label)));
            cplex.addGe(cplex.sum(cplex.prod(M2, cplex.diff(1, x[i][j])),
vert[j]), cplex.sum(t*d[i][j], cplex.prod(vert[i], (1 -
points.get(i).label))));
        }
    }
} // 误差传递
```

### 7.2.4 问题三蚁群算法代码

```
global a1 a2 b1 b2 c sita;
a1 = 25;
a2 = 15;
b1 = 20;
b2 = 25;
c = 30;
sita = 0.001;

q_0 = .4770;
numAnts = 50;
```

```
beta = -1;
maxIts = 1000;
numCities = 0;
Q = 0.1;
rho = .2;

params = struct('q_0', q_0, 'numAnts', numAnts, 'beta', beta,
'Q', Q, 'rho', rho);
ities = genCities('data1.txt');
numCities = size(cities, 2);
distance = [];
for i = 1 : numCities
    for j = 1 : numCities
        dis = [cities(i).x-cities(j).x, cities(i).y-cities(j).y,
cities(i).z-cities(j).z];
        distance(i, j) = norm(dis);
    end
end
tau = ones(numCities, numCities) / 5000;
bestPath = 1:numCities;
bestScore = [10000000000, 0];
fprintf('Initial Score: %f', bestScore);
numIts = 1;
scores = zeros(1, maxIts);

scoreM = [];
while numIts < maxIts;

    for ant_k = 1:numAnts

        path = [];
        currInd = 1;
        path(currInd) = 1;


        unvisited = 1:(numCities-1);
        unvisited(path(1)) = [];
        connectedvisited = [];
        connectedvisitedIndex = [];

        levelerror = 0;
        verterror = 0;
        isError = 0;
```

```
        for currInd = 1:(numCities - 1)
            r = path(currInd);
            connectedvisited = [];
            indexC = 1;

            if sita * distance(r, numCities) + levelerror <= c &&
sita * distance(r, numCities) + verterror <= c
                path(currInd + 1) = numCities;
                break;
            end
            for index = 1:size(unvisited, 2)
                if cities(unvisited(index)).label == 1 && sita *
distance(r, unvisited(index)) + levelerror <= a2 && sita *
distance(r, unvisited(index)) + verterror <= a1
                    connectedvisited(indexC) = unvisited(index);
                    connectedvisitedindex(indexC) = index;
                    indexC = indexC + 1;
                    continue;
                end
                if cities(unvisited(index)).label == 0 && sita *
distance(r, unvisited(index)) + levelerror <= b2 && sita *
distance(r, unvisited(index)) + verterror <= b1
                    connectedvisited(indexC) = unvisited(index);
                    connectedvisitedindex(indexC) = index;
                    indexC = indexC + 1;
                end
            end
            if (rand < q_0)
                [~, sInd] = max(tau(r, connectedvisited) .*
((distance(numCities, connectedvisited)).^beta));
                s = connectedvisited(sInd);
            else
                vec = tau(r, connectedvisited) .*
(distance(numCities, connectedvisited).^beta);
                probs = vec ./ sum(vec);

                if any(isnan(probs))
                    sInd = 1;
                else

                    sInd = find(cumsum(probs) > rand, 1);
                end
                s = connectedvisited(sInd);
            end
```

```
            if size(connectedvisited, 1) == 0
                isError = 1;
                break;
            end

            levelerror = sita * distance(r, s) + levelerror;
            verterror = sita * distance(r, s) + verterror;
            if cities(s).label == 1
                verterror = 0;
            else
                levelerror = 0;
            end

            path(currInd + 1) =  s;
            unvisited(connectedvisitedindex(sInd)) = [];
            if s == numCities
                break;
            end
        end
        if isError == 1
            continue;
        end
        [scoreM(1), scoreM(2)] = scorePath(path, cities,
distance);
        if  scoreM(2) > bestScore(2) + 0.001 || (scoreM(2) ==
bestScore(2) && scoreM(1) <= bestScore(1))
            bestScore = [scoreM(1), scoreM(2)]
            proResult = scoreM(2)
            bestPath = path;
        end

        toCities = circshift(path, [0,1]);

        for ind = 2:size(path, 2)
            fromCity = path(ind);
            toCity = toCities(ind);
            tau(fromCity, toCity) = tau(fromCity, toCity) + Q /
(scoreM(1));
        end
    end

    tau = (1 - rho) * tau;
```

```
    numIts = numIts + 1;
end
```

评价函数

```
function [distance, presult] = scorePath(cityPerm, cities,
distance)
global a1 a2 b1 b2 c sita;
pnode = [];
index = 1;
totalDistance = 0;
lengthPath = size(cityPerm, 2);
for i = 2 : lengthPath
    totalDistance = totalDistance + distance(cityPerm(i-1),
cityPerm(i));
    if cities(cityPerm(i)).success == 1
        pnode(index) = cityPerm(i);
        index = index + 1;
    end
end
if index < 10
    index = index - 1;
    p = ff2n(index);
    isError = zeros(pow2(index));
    presult = 0;
    for k = 1 : pow2(index)
        pp = p(k,:);
        ppindex = 1;
        levelError = 0;
        vertError = 0;
        pcount = 1;
        for i = 2 : lengthPath
            if ppindex <= index && cityPerm(i) == pnode(ppindex)
                if cities(cityPerm(i-1)).label == 1 && pp(ppindex)
== 1
                    levelError = levelError + distance(cityPerm(i-
1), cityPerm(i)) * sita;
                    vertError = min(5, vertError) +
distance(cityPerm(i-1), cityPerm(i)) * sita;
                    pcount = pcount * 0.2;
                end
                if cities(cityPerm(i-1)).label == 1 && pp(ppindex)
== 0
                    levelError = levelError + distance(cityPerm(i-
1), cityPerm(i)) * sita;
```

```
                vertError = distance(cityPerm(i-1), cityPerm(i))
* sita;
                pcount = pcount * 0.8;
            end
            if cities(cityPerm(i-1)).label == 0 && pp(ppindex)
== 1
                levelError = min(5, levelError) +
distance(cityPerm(i-1), cityPerm(i)) * sita;
                vertError = vertError + distance(cityPerm(i-1),
cityPerm(i)) * sita;
                pcount = pcount * 0.2;
            end
            if cities(cityPerm(i-1)).label == 0 && pp(ppindex)
== 0
                levelError = distance(cityPerm(i-1),
cityPerm(i)) * sita;
                vertError = vertError + distance(cityPerm(i-1),
cityPerm(i)) * sita;
                pcount = pcount * 0.8;
            end
            ppindex = ppindex + 1;
        else
            levelError = levelError * cities(cityPerm(i-
1)).label + distance(cityPerm(i-1), cityPerm(i)) * sita;
            vertError = vertError * (1 - cities(cityPerm(i-
1)).label) + distance(cityPerm(i-1), cityPerm(i)) * sita;
        end
        if cities(cityPerm(i)).label == 1
            if levelError > a2 || vertError > a1
                isError(k) = 1;
                break;
            end
        end
        if cities(cityPerm(i)).label == 0
            if levelError > b2 || vertError > b1
                isError(k) = 1;
                break;
            end
        end
        if cityPerm(i) == lengthPath
            if levelError > c || vertError > c
                isError(k) = 1;
                break;
            end
```

```
            end
        end
        if isError(k) == 0
            presult = presult + pcount;
        end
    end
else
    presult = 0.01;
end
distance = totalDistance;
end
```